$2.00
U.S.A.

S-100 MICROSYSTEMS™

JULY/AUG 1980 VOL. 1/NO. 4

A SUPER 8080 DEBUGGING EMULATOR

See Pages 22 – 31

Also in this Issue

*Registered trademark of Digital Research

and more

Complete Table of Contents on Page 3

26 MEGABYTES $4995.

DRIVE A HARD BARGAIN!

Suddenly, S-100 microcomputer systems can easily handle 100 million bytes. Because Morrow Designs™ now offers the first 26 megabyte hard disk memory for S-100 systems—the DISCUS M26™ Hard Disk System.

It has 26 megabytes of useable memory (29 megabytes unformatted). And it's expandable to 104 megabytes.

The DISCUS M26™ system is delivered complete—a 26 megabyte hard disk drive, controller, cables and operating system—for just $4995. Up to three additional drives can be added, $4495 apiece.

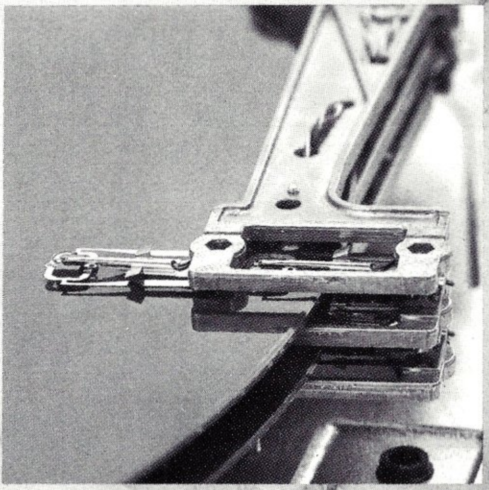
The DISCUS M26™ system features the Shugart SA4008 Winchester-type sealed media hard disk drive, in a handsome metal cabinet with fan and power supply.

The single-board S-100 controller incorporates intelligence to supervise all data transfers, communicating with the CPU via three I/O ports (command, status, and data). The controller has the ability to generate interrupts at the completion of each command to increase system throughput. There is a 512 byte sector buffer on-board. And each sector can be individually write-protected for data base security.

The operating system furnished with DISCUS M26™ systems is the widely accepted CP/M* 2.0.

See the biggest, most cost-efficient memory ever introduced for S-100 systems, now at your local computer shop. If unavailable locally, write Morrow Designs,™ 5221 Central Avenue, Richmond, CA 94804. Or call (415) 524-2101, weekdays 10-5 Pacific Time.

*CP/M is a trademark of Digital Research.



MORROW DESIGNS™
Thinker Toys™

S-100
MICROSYSTEMS
TM

Volume 1 Number 4 **July/August 1980**

Editorial Correspondence should be sent to: S-100 MICROSYSTEMS, BOX 1192, Mountainside, NJ 07092.

STAFF

Sol Libes
publisher/editor

Claudette Moore
managing editor

Patti Balinski
layout/typesetting

Jacob Epstein
CP/M* editor

Jon Bondy
Pascal editor

Don Libes
assistant editor

Lennie Libes
Susan Libes
subscriptions/office manager

S-100 MICROSYSTEMS is seeking articles on S-100 software, hardware and applications. Program listings should be typed on white paper with a new ribbon. Articles should be typed 40 characters/inch at 10 pitch. Author's name, address and phone number should be included on first page of article and all pages should be numbered. Photos are desirable and should be black and white glossy.

Commercial advertising is welcomed. Write to S-100 MICROSYSTEMS, 93 Washington St., Morristown, NJ 07960, or phone Claudette Moore at 201-267-4558.

*TMK Digital Research

IN THIS ISSUE

DEPARTMENTS

S-100 MICROSYSTEMS (USPS 529-530) is published six times per year for $9.50 per year (USA) by LIBES, INC., 995 Chimney Ridge, Springfield, NJ 07081. Controlled circulation pending at Baltimore, Maryland.
POSTMASTER: Send address changes to S-100 MICROSYSTEMS, PO Box 789-M, Morristown, NJ 07960.

The Editor's Page

by Sol Libes

The S-100 Bus: Past, Present And Future

Reprinted, with permission, from April 14, 1980 issue of INFOWORLD, 530 Lytton Ave., Palo Alto, CA 94301. Subscription $18/yr.

Part II

This is the second, and concluding article, analyzing the S-100 computer systems picture. The first part, which appeared in issue 3 covered the history of the S-100 bus systems. This article discusses its present status, and what I believe to be its future direction.

S-100 bus-based systems have been in manufacture now for over five years. The life span of the bus is the longest of any of the personal computer systems—over twice as long as that of the TRS-80, PET, and Apple computers. Moreover, there are more S-100 systems in operation than there are TRS-80, PET, and Apple computers; I would estimate that there are presently over 200,000 installed S-100 computer systems.

Furthermore, the dominance of the microcomputer field by S-100 systems will increase, since Radio Shack plans to stop production on the TRS-80, Model I before the end of the year. S-100's, I predict, will continue to be manufactured for a long time to come.

I have heard several people say, "S-100 is dead," and I have even seen statements like this in print. The articles that contained this statement usually included other erroneous statements as well. This misbelief is generally based on the fact that five S-100 manufacturers closed their doors in 1978-79 (although two subsequently opened again); however, the early demise of these companies was a result of mismanagement, not of their use of the S-100 bus. The fact is that during 1979, five new manufacturers of S-100 mainframes entered the market; so far this year, there are four new S-100 mainframe makers. I last counted a total of 19 S-100 mainframe manufacturers, over 60 manufacturers of plug-in boards, and a staggering 160 suppliers of software for S-100 systems.

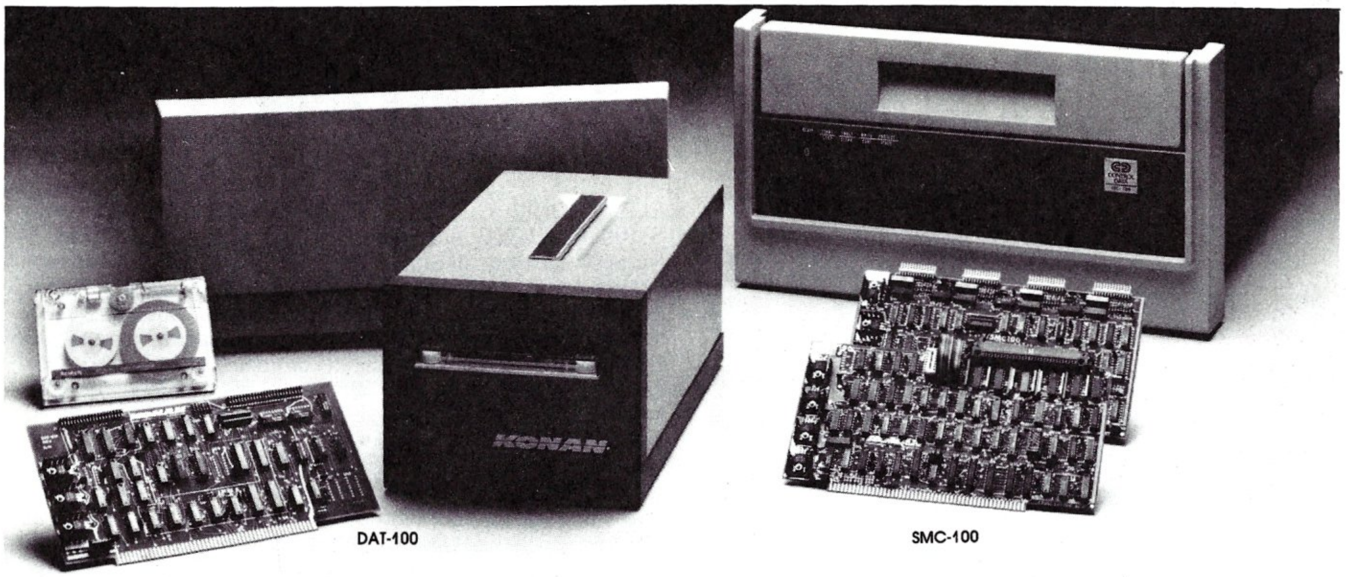
Since the number of manufacturers of S-100 products far exceed the number of manufacturers of products for TRS-80, Apple and PET, it is obvious that the gross S-100 business is greater.

The fact is, for sophisticated system development work, or for business or scientific applications, S-100 based systems are the only systems offering the necessary power and features. Furthermore, they provide these additional features at a cost that is competitive with the less powerful TRS-80, etc., systems. For example, try shopping for a word processing system to run a good word processor software package (e.g., WordStar), and you will find that an S-100 system is less expensive than a TRS-80 Model II (you cannot run WordStar on a Model I).

But, most importantly, S-100 based systems offer power, features, and flexibility that are just not available on most other personal systems. To be specific:

- More software is available. There are several times more high-level languages, operating systems, and applications packages for S-100 based systems than there are for any other system. Languages such as Basic, Fortran, Pascal, Cobol, APL, Algol, Lisp, C, and many more.
- There is greater computer power capability with the S-100. What other system has direct addressing of up to 16 M bytes of memory (24 address lines) and 64K input/output ports (16 address lines), up to eleven vectored interrupt lines, up to sixteen masters on the bus (with priority), up to twenty-three plug-in slots on the motherboard, up to 10 MHz clock on the bus, plug-in operator front panel, and more?
- The S-100 bus is now standardized by the Institute of Electrical and Electronic Engineers (IEEE), assuring conformity among manufacturers. The only other standardized bus is the Intel Multibus, which is less powerful and more expensive than the S-100.
- The modularity of the S-100 system assures that these systems can be upgraded with the changing state of the art. For example, there are owners of five-year-old Altairs who have installed 16-bit CPU's into their systems with just some plug-in board changes.

Today, the S-100 computer bus is a mature, refined system that has evolv-



Hard disk and hardtape™ control

Up to 2400 Megabytes of hard disk control for the S-100 bus.

Konan's SMC-100 interfaces S-100 bus micro computers with all hard disk drives having the Industry Standard SMD Interface. It is available with software drivers for most popular operating systems. Each SMC-100 controls up to 4 drives ranging from 8 to 600 megabytes per drive, including most "Winchester" drives -- such as Kennedy, Control Data, Fujitsu, Calcomp, Microdata, Memorex, Ampex, and others.

SMC-100 is a sophisticated, reliable system for transferring data at fast 6 to 10 megahertz rates with onboard sector buffering, sector interleaving, and DMA.

SMC-100's low cost-per-megabyte advanced technology keeps your micro computer system micro-priced. Excellent quantity discounts are available.

Konan's HARDTAPE™ subsystem . . . very low cost tape and/or hard disk Winchester backup and more.

Konan's new DAT-100 Single Board Controller interfaces with a 17½ megabyte (unformatted) cartridge tape drive as well as the Marksman Winchester disk drive by Century Data.

The DAT-100 "hardtape" system is the only logical way to provide backup for "Winchester" type hard disk systems. (Yields complete hard disk backup with data verification in 20-25 minutes.)

Konan's HARDTAPE™ subsystem is available off the shelf as a complete tape and disk mass storage system or an inexpensive tape and/or disk subsystem.

Konan controllers and subsystems support most popular software packages including FAMOS™, CP/M® version 2.X, and MP/M.

Konan, first (and still the leader) in high-reliability tape and disk mass storage devices, offers OEM's, dealers and other users continuing diagnostic support and strong warranties. Usual delivery is off the shelf to 30 days with complete subsystems on hand for immediate delivery.

Call Konan's TOLL FREE ORDER LINE today:

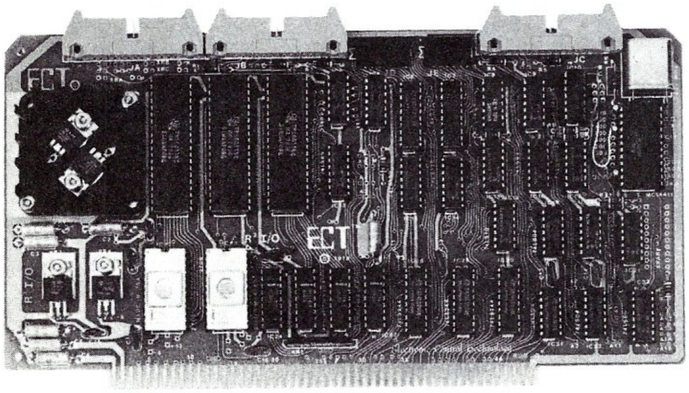
800-528-4563

Or write to Bob L. Gramley
Konan Corporation, 1448 N. 27th Avenue
Phoenix, AZ 85009. TWX/TELEX 9109511552

CP/M® is a registered trade name of Digital Research,
FAMOS™ is a trade name of MVT Micro Computer Systems.
HARDTAPE™ is a trade name of Konan Corporation.

KONAN

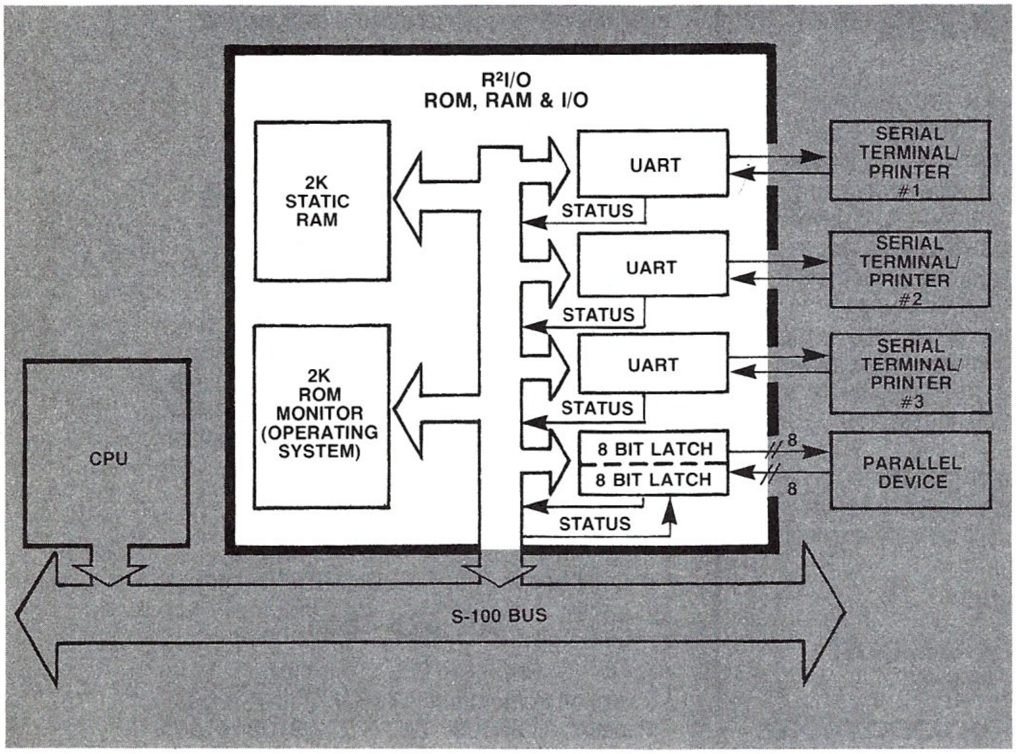
ECT R²I/O... The S-100 ROM, RAM & I/O Board



ELECTRONIC CONTROL TECHNOLOGY's R²I/O is an S-100 Bus I/O Board with 3 Serial I/O Ports (UART's), 1 Parallel I/O Port, 4 Status Ports, 2K of ROM with Monitor Program and 2K of Static RAM. The R²I/O provides a convenient means of interfacing several I/O devices, such as – CRT terminals, line printers, modems or other devices, to an S-100 Bus Microcomputer or dedicated controller. It also provides for convenient Microcomputer system control from a terminal keyboard with the 8080 Apple ROM monitor containing 26 Executive Commands and I/O routines. It can be used in dedicated control applications to produce a system with as few as two boards, since the R²I/O contains ROM, RAM and I/O.

The standard configuration has the Monitor ROM located at F000 Hex with the RAM at F800 Hex and the I/O occupies the first block of 8 ports. Jumper areas provide flexibility to change these locations, within reason, as well as allow the use of ROM's other than the 2708 (e.g. 2716 or similar 24 pin devices). Baud rates are individually selectable from 75 to 9600. Voltage levels of the Serial I/O Ports are RS-232.

- S-100 BUS
- 2K ROM
- 2K RAM
- 3 Serial I/O Ports
- 1 Parallel I/O Port
- 4 Status Ports
- ROM Monitor (Operating System)



8080 APPLE MONITOR COMMANDS

A – Assign I/O
B – Branch to user routine A-Z
C – Undefined
D – Display memory on console in Hex
E – End of file tag for Hex dumps
F – Fill memory with a constant
G – GOTO an address with breakpoints
H – Hex math sum & difference
I – User defined
J – Non-destructive memory test
K – User defined
L – Load a binary format file
M – Move memory block to another address
N – Nulls leader/trailer
O – User defined
P – Put ASCII into memory
Q – Query I/O ports: QI (N)-read I/O; QO(N,V)-send I/O
R – Read a Hex file with checksum
S – Substitute/examine memory in Hex
T – Types the contents of memory in ASCII equivalent
U – Unload memory in Binary format
V – Verify memory block against another memory block
W – Write a checksummed Hex file
X – Examine/modify CPU registers
Y – 'Yes there' search for 'N' Bytes in memory
Z – 'Z END' address of last R/W memory location

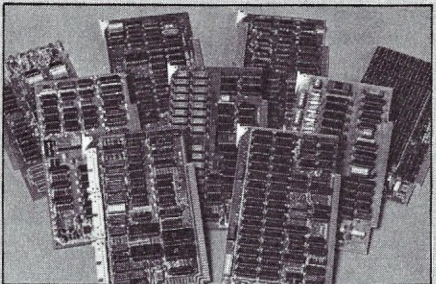
ECT™

Specializing in Quality Microcomputer Hardware
Building Blocks for Microcomputer Systems, Control and Test Equipment
Card Cages, Power Supplies, Mainframes, CPU's, Memory, I/O

ELECTRONIC CONTROL TECHNOLOGY **(201) 686-8080**
763 Ramsey Ave., Hillside, N.J. 07205

At Intersystems, "dump" is an instruction. Not a way of life.

(Or, when you're ready for IEEE S-100, will your computer be ready for you?)



We're about to be gadflies again.

While everyone's been busy trying to convince you that large buses housed in strong metal boxes will guarantee versatility and ward off obsolescence, we've been busy with something better. Solving the *real* problem with the first line of computer products *built from the ground up to conform to the new IEEE S-100 Bus Standard*. Offering you extra versatility in 8-bit applications today. And a full 16 bits tomorrow.

We call our new line Series II.™ And even if you don't need the full 24-bit address for up to 16 megabytes (!) of memory right now, they're something to think about. Because of all the performance, flexibility and economy they offer. Whether you're looking at a new mainframe, expanding your present one or upgrading your system with an eye to the future. (Series II boards are compatible with most existing S-100 systems and *all* IEEE S-100 Standard cards as other manufacturers get around to building them.)

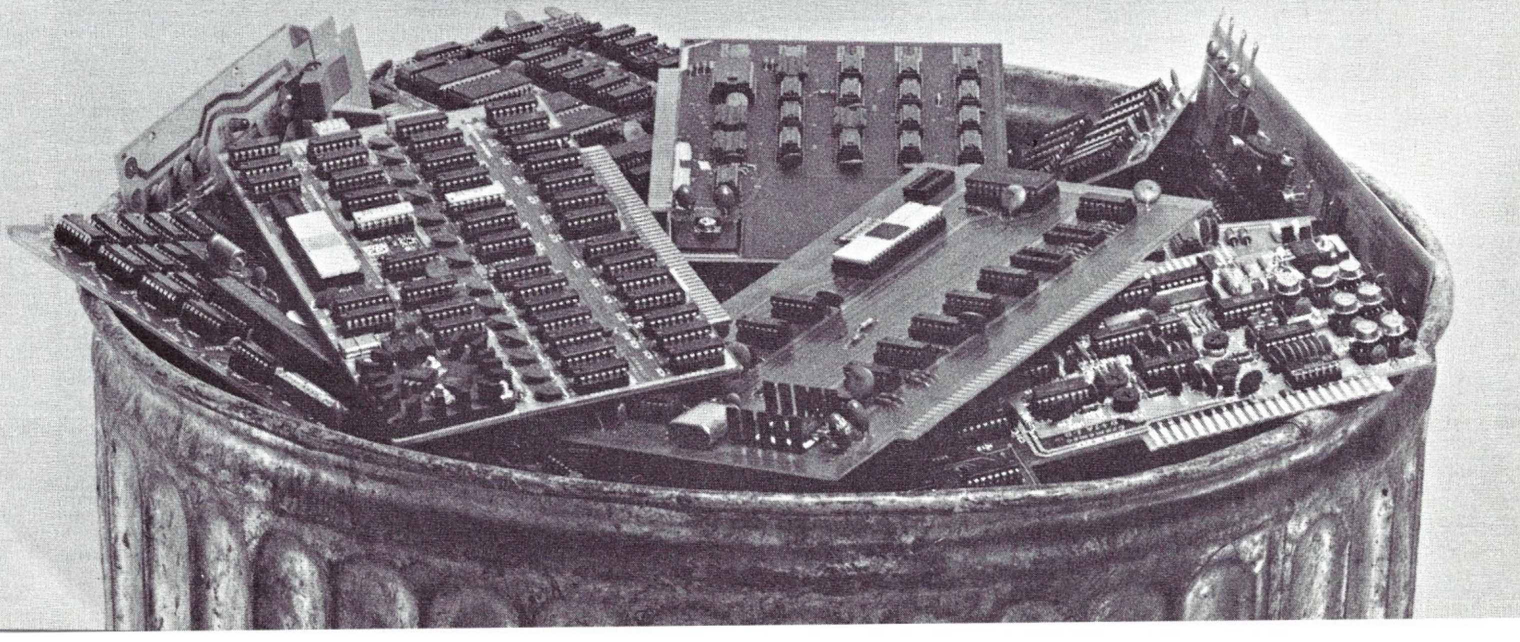
Consider some of the features: Reliable operation to 4MHz and beyond. Full compatibility with 8- and 16-bit CPUs, peripherals and other devices. *Eight* levels of prioritized interrupts. Up to 16 individually-addressable DMA devices, with IEEE Standard overlapped operation. User-selectable functions addressed by DIP-switch or jumpers, eliminating soldering. And that's just for openers.

The best part is that all this heady stuff is available *now*! In our advanced processor—a full IEEE Bus Master featuring Memory Map™ addressing to a full megabyte. Our fast, flexible 16K Static RAM and 64K Dynamic RAM boards. An incredibly versatile and economical 2-serial, 4-parallel Multiple I/O board. 8-bit A/D-D/A converter. Our Double-Density High-Speed Disk Controller. And what is undoubtedly the most flexible front panel in the business. Everything you need for a complete IEEE S-100 system. Available separately, or all together in our new DPS-1 Mainframe!

Whatever your needs, why dump your money into obsolete products labelled "IEEE timing compatible" or other words people use to make up for a lack of product. See the future now, at your Intersystems dealer or call/write for our new catalog. We'll tell you all about Series II and the new IEEE S-100 Bus we helped pioneer. Because it doesn't make sense to buy yesterday's products when tomorrow's are already here.

InterSystems™

Ithaca Intersystems Inc.,
1650 Hanshaw Road/P.O. Box 91,
Ithaca, NY 14850
607-257-0190/TWX: 510 255 4346



No. 14: Take a byte.

Software for most popular 8080/Z80 computer disk systems including*
***NORTH STAR, iCOM, MICROPOLIS, DYNABYTE DB8/2 & DB8/4, EXIDY SORCERER, SD SYSTEMS, ALTAIR, VECTOR MZ, MECA, 8" IBM, HEATH H17 & H89, HELIOS, IMSAI VDP42 & 44, REX, NYLAC, INTERTEC SUPER-BRAIN, VISTA V80 and V200, TRS-80* MODEL I and MODEL II, ALTOS, OHIO SCIENTIFIC, DIGI-LOG, KONTRON PSI-80, IMS 5000** diskette formats and **CSSN BACKUP** cartridge tapes.*

Genuine CP/M for Apple II coming soon! Call for details.

Prices reflect distribution on 8" single density diskettes. If a format is requested which requires additional diskettes, a surcharge of $8. per additional diskette will be added. A surcharge of $25 will be added for software on CSSN format DC 300XL cartridges. Media charge for 5440 disk is $100.

CP/M® VERSION 2 FOR TRS-80 MODEL II NOW AVAILABLE

All Lifeboat programs require CP/M, unless otherwise stated.

Software with Manual / Manual Alone

□ **CP/M* FLOPPY DISK OPERATING SYSTEM** — Digital Research's operating system configured for many popular micro-computers and disk systems:

System	Version	Price
North Star Single Density	1.4	**145/25** v
North Star Double Density	1.4	**145/25**
North Star Double/Quad	2.x	**170/25**
iCOM Micro-Disk 2411	1.4	**145/25**
iCOM 3712	1.4	**170/25** v*
iCOM 3812	1.4	**170/25** *
Mits 3202/Altair 8800	1.4	**145/25**
Heath H8 + H17	1.4	**145/25** Ⓜ
Heath H89	1.4	**145/25** Ⓜ
Heath H89 by Magnolia	1.4	**250/25**◊
TRS-80 Model I	1.4	**145/25** Ⓜ
TRS-80 Model II	2.x	**170/25**
TRS-80 Model II + Corvus	2.x	**250/25**
Processor Technology Helios II	1.4	**145/25**
Cromemco System 3	1.4	**145/25**
Intel MDS Single Density	1.4	**145/25**
Intel MDS Single Density	2.x	**170/25**
Micropolis Mod I	1.4	**145/25** v
Micropolis Mod II	1.4	**145/25** v

The following configurations are scheduled for release soon:

System	Version	Price
Apple II	2.x	**350/25**◊
North Star Double/Quad + Corvus	2.x	**250/25**
North Star Horizon HD-1	2.x	**250/25**
Ohio Scientific C3	2.x	**200/25**
Ohio Scientific C3-C	2.x	**250/25**
Micropolis Mod II	2.x	**200/25**
Mostek MDX STD Bus System	2.x	**350/25** **
iCOM 3812	2.x	**225/25** *
iCOM 4511/Pertec D3000	2.x	**375/25** * +
Durango F-85	2.x	**170/25**

*Software consists of the operating system, text editor, assembler, debugger and other utilities for file management and system maintenance. Complete set of Digital Research's documentation and additional implementation notes included. Systems marked * and ** include firmware on 2708 and 2716. Systems marked + include 5440 media charge. Systems marked Ⓜ require the special Ⓜ versions of software in this catalog. Systems marked v have minor variants available to suit console interface of system. Call or write for full list of options. ◊ includes hardware addition to allow our standard versions of software to run under it.*

□ Ⓜ **Z80 DEVELOPMENT PACKAGE** — Consists of: (1) disk file line editor, with global inter and intra-line facilities; (2) Z80 relocating assembler, Zilog/Mostek mnemonics, conditional assembly and cross reference table capabilities; (3) linking loader producing absolute Intel hex disk file .... **$95/$20**

□ Ⓜ **ZDT** — Z80 Monitor Debugger to break and examine registers with standard Zilog/Mostek mnemonic disassembly displays. $35 when ordered with Z80 Development Package .... **$50/$10**

□ **XASM-68** — Non-macro cross-assembler with nested conditionals and full range of pseudo operations. Assembles from standard Motorola MC6800 mnemonics to Intel hex .... **$200/$25**

□ **XASM-65** — As XASM-68 for MOS Technology MCS-6500 series mnemonics .... **$200/$25**

□ **DISTEL** — Disk based disassembler to Intel 8080 or TDL/Xitan Z80 source code, listing and cross reference files, Intel or TDL/Xitan pseudo ops optional. Runs on 8080 .... **$65/$10**

□ Ⓜ **DISILOG** — As DISTEL to Zilog/Mostek mnemonic files. Runs on Z80 only .... **$65/$10**

□ Ⓧ **SMAL/80** Structured Macro Assembler Language — Package of powerful general purpose text macro processor and SMAL structured language compiler. SMAL is an assembler language with IF-THEN-ELSE, LOOP-REPEAT-WHILE, DO-END, BEGIN-END constructs .... **$75/$15**

□ Ⓧ **tiny C** — Interactive interpretive system for teaching structured programming techniques. Manual includes full source listings .... **$105/$50**

New Features

□ Ⓜ ⓘ **BDS C COMPILER** — Supports most features of language, including Structures, Arrays, Pointers, recursive function evaluation, overlays. Includes linking loader, library manager, and library containing general purpose, file I/O, and floating point functions. Lacks initializers, statics, floats and longs. Documentation includes "The C PROGRAMMING LANGUAGE" by Kernighan and Ritchie .... **$125/$20**

□ Ⓛ Ⓧ **WHITESMITHS C COMPILER** — The ultimate in systems software tools. Produces faster code than a pseudo-code Pascal with more extensive facilities. Conforms to the full UNIX* Version 7 C language, described by Kernighan and Ritchie, and makes available over 75 functions for performing I/O, string manipulation and storage allocation. Linkable to Microsoft REL files. Requires 60K CP/M .... **$630/$30**

Prices and specifications subject to change without notice.

Software with Manual / Manual Alone

MICROSOFT

□ Ⓛ Ⓜ **BASIC-80** — Disk Extended BASIC, ANSI compatible with long variable names, WHILE/WEND, chaining, variable length file records .... **$325/$25**

□ Ⓛ Ⓜ **BASIC COMPILER** — Language compatible with BASIC-80 and 3-10 times faster execution. Produces standard Microsoft relocatable binary output. Includes MACRO-80. Also linkable to FORTRAN-80 or COBOL-80 code modules .... **$350/$25**

□ Ⓛ Ⓜ **FORTRAN-80** — ANSI 66 (except for COMPLEX) plus many extensions. Includes relocatable object compiler, linking loader, library with manager. Also includes MACRO-80 (see below) .... **$425/$25**

□ Ⓛ Ⓜ **COBOL-80** — Level 1 ANSI '74 standard COBOL plus most of Level 2. Full sequential, relative, and indexed file support with variable file names. STRING, UNSTRING, COMPUTE, VARYING/UNTIL, EXTEND, CALL, COPY, SEARCH, 3-dimensional arrays, compound and abbreviated conditions, nested IF. Powerful interactive screen-handling extensions. Includes compatible assembler, linking loader, and relocatable library manager as described under MACRO-80 .... **$700/$25**

□ Ⓛ Ⓜ **MACRO-80** — 8080/Z80 Macro Assembler. Intel and Zilog mnemonics supported. Relocatable linkable output. Loader, Library Manager and Cross Reference List utilities included .... **$149/$15**

□ Ⓛ **XMACRO-86** — 8086 cross assembler. All Macro and utility features of MACRO-80 package. Mnemonics slightly modified from Intel ASM86. Compatibility data sheet available .... **$275/$25**

□ Ⓛ **EDIT-80** — Very fast random access text editor for text with or without line numbers. Global and intra-line commands supported. File compare utility included. .... **$89/$15**

New Features

□ **PASCAL/M*** — Compiles enhanced Standard Pascal to compressed efficient Pcode. Totally CP/M compatible. Random access files. Both 16 and 32-bit Integers. Runtime error recovery. Convenient STRINGs. OTHERWISE clause on CASE. Comprehensive manual (90 pp, indexed). SEGMENT provides overlay structure. INPORT, OUTPORT and untyped files for arbitrary I/O. Requires 56K CP/M .... **$175/$20**

□ ⓘ **PASCAL/Z** — Z80 native code PASCAL compiler. Produces optimized, ROMable re-entrant code. All interfacing to CP/M is through the support library. The package includes compiler, relocating assembler and linker, and source for all library modules. Variant records, strings and direct I/O are supported. Requires 56K CP/M and Z80 CPU .... **$395/$25**

New Version New Features

□ Ⓜ Ⓧ **PASCAL/MT** — Subset of standard PASCAL. Generates ROMable 8080 machine code. Symbolic debugger included. Supports interrupt procedures, CP/M file I/O and assembly language interface. Real variables can be BCD, software floating point, or AMD 9511 hardware floating point. Includes strings enumerations and record data types. Manual explains BASIC to PASCAL conversion. Requires 32K. .... **$250/$30**

□ **ALGOL-60** — Powerful block-structured language compiler featuring economical run-time dynamic allocation of memory. Very compact (24K total RAM) system implementing almost all Algol 60 report features plus many powerful extensions including string handling direct disk address I/O etc. Requires Z80 CPU .... **$199/$20**

□ Ⓜ **CBASIC-2** Disk Extended BASIC — Non-interactive BASIC with pseudo-code compiler and run-time interpreter. Supports full file control, chaining, integer and extended precision variables, etc. ..... **$120/$15**

MICRO FOCUS

□ Ⓛ Ⓜ **STANDARD CIS COBOL** — ANSI '74 COBOL standard compiler fully validated by U.S. Navy tests to ANSI level 1. Supports many features to level 2 including dynamic loading of COBOL modules and a full ISAM file facility. Also, program segmentation, interactive debug and powerful interactive extensions to support protected and unprotected CRT screen formatting from COBOL programs used with any dumb terminal .... **$850/$50**

□ Ⓛ **FORMS 2** — CRT screen editor. Output is COBOL data descriptions for copying into CIS COBOL programs. Automatically creates a query and update program of indexed files using CRT protected and unprotected screen formats. No programming experience needed. Output program directly compiled by STANDARD CIS COBOL .... **$200/$20**

Tasty lower prices!

EIDOS SYSTEMS

□ Ⓛ **KISS** — Keyed Index Sequential Search. Offers complete Multi-Keyed Index Sequential and Direct Access file management. Includes built-in utility functions for 16 or 32 bit arithmetic, string/integer conversion and string compare. Delivered as a relocatable linkable module in Microsoft format for use with FORTRAN-80 or COBOL-80, etc. .... **$335/$23**

Software with Manual / Manual Alone

□ Ⓛ **KBASIC** -- Microsoft Disk Extended BASIC version 4.51 integrated by implementation of nine additional commands in language. Package includes KISS.REL as described above, and a sample mail list program. .... **$585/$45**
To licensed users of Microsoft BASIC-80 (MBASIC) .... **$435/$45**

□ **XYBASIC** Interactive Process Control BASIC — Full disk BASIC features plus unique commands to handle bytes, rotate and shift, and to test and set bits. Available in integer and extended versions.
Interger ROMable, ROM Squared and CP/M **$295/$25**
Extended ROMable, ROM Squared and CP/M **$395/$25**
Extended disk XYBASIC for CP/M .... **$495/$25**

NEW

□ **RECLAIM** — A utility to validate media under CP/M. Program tests a diskette or hard disk surface for errors, reserving the imperfections in invisible files, and permitting continued usage of the remainder. Essential for any hard disk. Requires CP/M version 2. .... **$80/$5**

□ Ⓜ **BASIC UTILITY DISK** — Consists of: (1) CRUNCH-14 — Compacting utility to reduce the size and increase the speed of programs in Microsoft BASIC 4.51, BASIC-80 and TRS-80 BASIC. (2) DPFUN — Double precision subroutines for computing nineteen transcendental functions including square root, natural log, log base 10, sine, arc sine, hyperbolic sine, hyperbolic arc sine, etc. Furnished in source on diskette and documentation .... **$50/$35**

□ **STRING/80** — Character string handling plus routines for direct CP/M BDOS calls from FORTRAN and other compatible Microsoft languages. The utility library contains routines that enable programs to chain to a COM file, retrieve command line parameters, and search file directories with full wild card facilities. Supplied as linkable modules in Microsoft format. .... **$95/$20**

□ **STRING/80** source code available separately **$295/NA**

□ Ⓜ **THE STRING BIT** — FORTRAN character string handling. Routines to find, fill, pack, move, separate, concatenate and compare character strings. This package completely eliminates the problems associated with character string handling in FORTRAN. Supplied with source .... **$65/$15**

□ Ⓜ **VSORT** — Versatile sort/merge system for fixed length records with fixed or variable length fields. VSORT can be used as a stand-alone package or loaded and called as a subroutine from CBASIC-2. When used as a subroutine, VSORT maximizes the use of buffer space by saving the TPA on disk and restoring it on completion of sorting. Records may be up to 255 bytes long with a maximum of 5 fields. Upper/lower case translation and numeric fields supported. .... **$175/$20**

□ **CPM/374X** — Has full range of functions to create or re-name an IBM 3741 volume, display directory information and edit the data set contents. Provides full file transfer facilities between 3741 volume data sets and CP/M files .... **$195/$10**

□ Ⓜ **BSTAM** — Utility to link one computer to another also equipped with BSTAM. Allows file transfers at full data speed (no conversion to hex), with CRC block control check for very reliable error detection and automatic retry. We use it! It's great! Full wildcard expansion to send ★.COM, etc. 9600 baud with wire. 300 baud with phone connection. **Both** ends need one. Standard and Ⓜ versions can talk to one another. .... **$150/$10**

□ **WHATSIT?*** Interactive data-base system using associative tags to retrieve information by subject. Hashing and random access used for fast response. Requires CBASIC-2 .... **$175/$25**

□ Ⓜ † **SELECTOR III-C2** — Data Base Processor to create and maintain multi Key data bases. Prints formatted sorted reports with numerical summaries or mailing labels. Comes with sample applications, including Sales Activity, Inventory, Payables, Receivables, Check Register, and Client/Patient Appointments, etc. Requires CBASIC-2. Supplied in source ... **$295/$20**

□ **GLECTOR** — General Ledger option to SELECTOR III-C2. Interactive system provides for customized COA. Unique chart of transaction types insure proper double entry bookkeeping. Generates balance sheets, P&L statements and journals. Two year record allows for statement of changes in financial position report. Supplied in source. Requires SELECTOR III-C2, CBASIC-2 and 56K system .... **$350/$25**

□ † **CBS** — Configurable Business System is a comprehensive set of programs for defining custom data files and application systems without using a programming language such as BASIC, FORTRAN, etc. Multiple key fields for each data file are supported. Set-up program customizes system to user's CRT and printer. Provides fast and easy interactive data entry and retrieval **with transaction processing.** Report generator program does complex calculations with stored and derived data, record selection with multiple criteria, and custom formats. Sample inventory and mailing list systems included. **No support language required** .... **$295/$40**

Software with Manual / Manual Alone

all Micropro prices are discounted!

MICROPRO

□ Ⓛ **SUPER-SORT I** — Sort, merge, extract utility as absolute executable program or linkable module in Microsoft format. Sorts fixed or variable records with data in binary, BCD, Packed Decimal, EBCDIC, ASCII, floating & fixed point, exponential, field justified, etc. Even variable number of fields per record! . **$225/$25**

□ Ⓛ **SUPER-SORT II** — Above available as absolute program only .... **$175/$25**

□ Ⓛ **SUPER-SORT III** — As II without SELECT/EXCLUDE .... **$125/$25**

□ Ⓛ **WORD-STAR** — Menu driven visual word processing system for use with standard terminals. Text formatting performed on screen. Facilities for text paginate, page number, justify, center and underscore. User can print one document while simultaneously editing a second. Edit facilities include global search and replace. Read/Write to other text files, block move, etc. Requires CRT terminal with addressable cursor positioning .... **$445/$40**

□ **WORD-STAR Customization Notes** — For sophisticated users who do not have one of the many standard terminal or printer configurations in the distribution version of WORD-STAR .... **NA/$95**

□ Ⓛ **WORD-MASTER** Text Editor — In one mode has superset of CP/M's ED commands including global searching and replacing, forwards and backwards in file in video mode, provides full screen editor for users with serial addressable-cursor terminal .... **$145/$25**

□ Ⓜ **TEXTWRITER III** — Text formatter to justify and paginate letters and other documents. Special features include insertion of text during execution from other disk files or console, permitting recipe documents to be created from linked fragments on other files. Has facilities for sorted index, table of contents and footnote insertions. Ideal for contracts, manuals, etc. Now compatible with Electric Pencil* prepared files. .... **$125/$20**

Now-applications software for Microsoft's BASIC interpreter!

PEACHTREE SOFTWARE

□ Ⓛ † **GENERAL LEDGER** — Records details of all financial transactions. Generates a balance sheet and an income statement. Flexible and adaptable design for both small businesses and firms performing client writeup services. Produces reports as follows: Trial Balance, Transaction Registers, Balance Sheet, Prior Year Comparative Balance Sheet, Income Statement, Prior Year Comparative Income Statement and Department Income Statements. Interactive with other PEACHTREE accounting packages. Supplied in source code for Microsoft BASIC .... **$990/$30**

□ Ⓛ † **ACCOUNTS PAYABLE** — Tracks current and aged payables and incorporates a check writing feature. Maintains a complete vendor file with information on purchase orders and discount terms as well as active account status. Produces reports as follows: Open Voucher Report, Accounts Payable Aging Report and Cash Requirements. Provides input to PEACHTREE General Ledger. Supplied in source code for Microsoft BASIC .... **$990/$30**

□ Ⓛ † **ACCOUNTS RECEIVABLE** — Generates invoice register and complete monthly statements. Tracks current and aged receivables. Maintains customer file including credit information and account status. The current status of any customer account is instantly available. Produces reports as follows: Aged Accounts Receivable, Invoice Register, Payment and Adjustment Register and Customer Account Status Report. Provides input to PEACHTREE General Ledger. Supplied in source code for Microsoft BASIC .. **$990/$30**

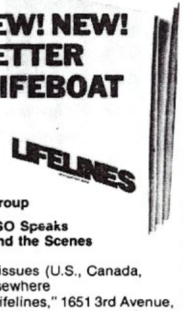
□ Ⓛ † **PAYROLL** — Prepares payroll for hourly, salaried and commissioned employees. Generates monthly, quarterly and annual returns. Prepares employee W-2's. Includes tables for federal withholding and FICA as well as withholding for all 50 states plus up to 20 cities from pre-computed or user generated tables. Will print checks, Payroll Register, Monthly Summary and Unemployment Tax Report. Provides input to PEACHTREE General Ledger. Supplied in source code for Microsoft BASIC .... **$990/$30**

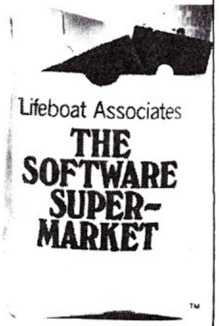
□ Ⓛ † **INVENTORY** — Maintains detailed information on each inventory item including part number, description, unit of measure, vendor and reorder data, item activity and complete information on current item costs, pricing and sales. Produces reports as follows: Physical Inventory Worksheet, Inventory Price List, Departmental Summary Report, Inventory Status Report, The Reorder Report and the Period-to-Date and Year-to-Date reports. Supplied in source code for Microsoft BASIC .... **$1190/$30**

New Address   New Phone

Lifeboat Associates, 1651 Third Avenue, N.Y., N.Y. 10028 **(212) 860-0300** Telex: 220501

™The Software Supermarket is a trademark of Lifeboat Associates





Ed. Page, cont'd...

ed into a professional-level computer system. It is, therefore, seeing increased use in the industrial, commercial, scientific, and small business areas. Further, because of its low cost and the availability of much S-100 equipment in kit form, it is very popular with computer hobbyists.

The S-100 computer picture will be one of steady, increased growth, albeit not the spectacular growth of 1975 through 1977, when we saw sales double each year. S-100 will compete more and more intensely with minicomputer systems in the markets that, until now, were exclusively the domain of the minicomputer.

Because of their low cost/high power ratio, S-100 computer systems will continue to be popular in the hobby and personal computer area. However, the consumer products manufacturers are moving into the personal computer market with new, very low-cost systems that, to a large extent, (will displace S-100 from these markets.

S-100 systems will increase and improve in power, flexibility, speed, and reliability. This trend has already begun, as is evidenced by the new 16-bit CPU cards and co-processor cards now being introduced. Today, typical memory size for an S-100 system is in the 48K-64K range. The likelihood is that next year, it will be in the 64K-128K range, and in 1983, it will typically be 128-256K.

Further, S-100 speed can be expected to increase dramatically. Today, most S-100 users employ 2-4MHz processor clocking. By 1981, this should typically be 6 MHz, and as high as 12MHz by 1983. Coupled with the more powerful 16-bit microprocessors, this means that S-100 microcomputers will take over the traditional minicomputer market by the mid 1980's. ■

Attention S-100 Board Suppliers

We are currently attempting to compile a listing of all S-100 board and mainframe suppliers and their products. If you are a manufacturer of S-100 products please send us a complete set of specification sheets on your products. We hope to publish this listing in the NOV/DEC issue of S-100 MICROSYSTEMS.

LETTERS TO THE EDITOR

Dear Editor:

After reading the first two issues of "S-100 MICROSYSTEMS" I was convinced that I had to subscribe. Here in Europe it is very difficult to get any information on computers except by reading American magazines. Even worse is the situation if you want to talk to someone having another S-100 System. My nearest "neighbor" lives 300 miles away.

For those people running CP/M it must be a challenge to write some interesting articles and submit them on disk! You ought to list those formats you will be able to read!

Will there be "classified" ad's? Hoping to find other people in Europe to talk to, and with best wishes for the success of this publication.

Here, in the north of Germany, we have a small club of thirty people meeting once a month for a sort of a "Computer Club." There are 3 who use CP/M (one OSI, one Cromemco, and me) and 2 who use an S-100 System. I know of two professional installations of a Cromemco and one other CP/M installation with an AMD-Distributor. Besides that, I know of three other Ham's in south-Germany using CP/M either on an ALTAIR or an ALTOS.

Holger Petersen
West Germany

We will accept "classifieds" of a non-commercial nature free of charge. Authors can submit articles on either 8" CP/M disk or 5-1/4" North Star CP/M. Text should be entered using either "Wordstar" or "Electric Pencil." We have an author's guide available. —Ed.

Dear Editor:

I completely agree with you that the S-100 bus is the best supported computer bus around. In the first issue of your fine magazine, you stated that there were seven different 8-bit processors already interfaced to the S-100 bus (8080, 8085, 8088, Z80 6502, 6800, and 6809). I found this comment amusing because it is such an understatement.

One interesting S-100 microprocessor you did not mention is the Signetics 2650. Several years ago, Victoria Micro Digital (whose address is 401 Dundee Street, Victoria, Texas 77901) implemented the Signetics 2650 onto the S-100 bus in a very interesting and unusual way. Their product is called the Slavemaster 2650 Multiprocessor, which consists of two separate S-100 cards. Each card is identical and each contains a 2650 mpu, Kansas City cassette interface, serial I/O, 8 vectored interrupts, a real-time clock interrupt, keyboard interrupt, AC power-fail interrupt, and four 2708 sockets. The two cards connect together via a 16-pin dip-plug ribbon cable. Either card can be jumpered to be the slave or the master. In multiprocessor mode, both processors run at full speed. This is accomplished by synchronizing circuitry that causes the two processors to interleave S-100 execute and fetch cycles. There are two memory modes. One is the common memory mode, where both processors can execute anywhere in memory. The other is split-memory mode, where each processor is restricted to its own 32K block. In split-memory mode, there is a feature called the mailbox, which allows each processor to access a 1K block of the other processor's memory. I found these boards to be a fascinating addition to my S-100 collection. However, many people may not be interested in anything as exotic as multiprocessing and may find more useful the fact that with just one of these cards the user has a powerful standalone 2650 S-100 bus microcomputer. The boards are of the high quality I've come to expect (and demand) of S-100 products. They're solder-masked, silkscreened, and has a gold-plated edge connector. Schematics are included and the write-up is pretty good. The blank boards (which is the way I normally prefer to buy S-100 boards) cost me $49 a piece. The price of a 1-board kit is $139 and is $189 assembled. I should add that because this product was designed a few years ago, it probably won't work with dynamic memory.

Another microprocessor you did not mention is the Motorola 6802. MicroDaSys (pronounced micro-daisies, as in the flower) successfully interfaced this processor to the S-100 bus two years ago. Their address is 357 South Lorraine Blvd, Los Angeles, CA 90020. I paid $50 for a blank board. The board is, of course, silk-screened, solder-masked, and gold-plated. However, the documentation is only fair. This board has a number of features. It has five 2708/2716 sockets, a serial interface, two bi-directional parallel ports. 1K bytes of static RAM, and a Kansas City cassette interface. The kit price is $200 and is $258 assembled. For $40 more, this board is up-gradable to Motorola's new 6809 processor. Because this board was designed Before the establishment of the IEEE S-100 standards, it probably will not work with dynamic memory. However, both the Victoria Micro Digital 2650 Slavemaster and MicroDaSys 6802/6809 will work fine with most of the static memory boards currently on the market.

Kenneth Young
Los Angeles, CA

Dear Editor:

I'd like to invite interested readers to attend upcoming meetings of the GREATER NY SOL USER'S GROUP. We meet at 7:30 on the second Monday of the month at the Westchester Federal Savings Bank, 67 Purchase Street, Rye, NY.

In addition to a SOL, most of our members have North Star minifloppy drives, some have additional 8" drives. Our interest ranges from keeping the SOL alive, to software and peripheral equipment. Applications range from business related programs, to word processing and of course—games.

We are an active club. If you know of anyone who would like to give a talk to the group we would be happy to discuss the details.

Best wishes on the success of S-100 MICROSYSTEMS.

Andre McHose
Ridgefield, CT

Dear Editor:

I am puzzled by the video display boards available for the S-100 bus. One can buy off the shelf any number of fully assembled ready to run boards that implement all the functions of a CRT terminal--except for the keyboard. I am unable to locate anywhere a completely packaged, ready to use by the consumer, keyboard complete with cabling necessary to work with a video board. What keyboards there are seem to be custom assemblies by local computer stores acting as OEM's. Thus while the video board portion of a terminal is backed by factory service and warranty, such a board does not supply a terminal with the same convenience. I believe this explains a great deal of the attractiveness of TRS-80, APPLE and similar systems. The practice of operating S-100 systems through standard serial data terminals is not just more expensive than using an integrated unit, it is logically unnecessary, a kind of kluge; the production of so many different video boards atests to that. The great popularity of the now discontinued SOL computer further illustrates this. But what's the point of all the video boards if no keyboards are supplied with them?

George Lyons
Jersey City, NJ

NEW! TPM* for TRS-80 Model II
NEW! System/6 Package
Computer Design Labs

Z80* Disk Software

We have acquired the rights to all TDL software (& hardware). TDL software has long had the reputation of being the best in the industry. Computer Design Labs will continue to maintain, evolve and add to this superior line of quality software.

— Carl Galletti and Roger Amidon, owners.

Software with Manual/Manual Alone

All of the software below is available on any of the following media for operation with a Z80 CPU using the CP/M* or similar type disk operating system (such as our own TPM*).

for TRS-80* CP/M (Model I or II)
for 8" CP/M (soft sectored single density)
for 5¼" CP/M (soft sectored single density)
for 5¼" North Star CP/M (single density)
for 5¼" North Star CP/M (double density)

BASIC I

A powerful and fast Z80 Basic interpreter with EDIT, RENUMBER, TRACE, PRINT USING, assembly language subroutine CALL, LOADGO for "chaining", COPY to move text, EXCHANGE, KILL, LINE INPUT, error intercept, sequential file handling in both ASCII and binary formats, and much, much more. It runs in a little over 12 K. An excellent choice for games since the precision was limited to 7 digits in order to make it one of the fastest around. $49.95/$15.

BASIC II

Basic I but with 12 digit precision to make its power available to the business world with only a slight sacrifice in speed. Still runs faster than most other Basics (even those with much less precision). $99.95/$15.

BUSINESS BASIC

The most powerful Basic for business applications. It adds to Basic II with random or sequential disk files in either fixed or variable record lengths, simultaneous access to multiple disk files, PRIVACY command to prohibit user access to source code, global editing, added math functions, and disk file maintenance capability without leaving Basic (list, rename, or delete). $179.95/$25.

ZEDIT

A character oriented text editor with 26 commands and "macro" capability for stringing multiple commands together. Included are a complete array of character move, add, delete, and display function. $49.95./$15.

ZTEL

Z80 Text Editing Language - Not just a text editor. Actually a language which allows you to edit text and also write, save, and recall programs which manipulate text. Commands include conditional branching, subroutine calls, iteration, block move, expression evaluation, and much more. Contains 36 value registers and 10 text registers. Be creative! Manipulate text with commands you write using Ztel. $79.95/$25.

TOP

A Z80 Text Output Processor which will do text formatting for manuals, documents, and other word processing jobs. Works with any text editor. Does justification, page numbering and headings, spacing, centering, and much more! $79.95/$25.

MACRO I

A macro assembler which will generate relocateable or absolute code for the 8080 or Z80 using standard Intel mnemonics plus TDL/Z80 extensions. Functions include 14 conditionals, 16 listing controls, 54 pseudo-ops, 11 arithmetic/logical operations, local and global symbols, chaining files, linking capability with optional linker, and recursive/reiterative macros. This assembler is so powerful you'll think it is doing all the work for you. It actually makes assembly language programming much less of an effort and more creative. $79.95/$20.

MACRO II

Expands upon Macro I's linking capability (which is useful but somewhat limited) thereby being able to take full advantage of the optional Linker. Also a time and date function has been added and the listing capability improved. $99.95/$25.

LINKER

How many times have you written the same subroutine in each new program? Top notch professional programmers compile a library of these subroutines and use a Linker to tie them together at assembly time. Development time is thus drastically reduced and becomes comparable to writing in a high level language but with all the speed of assembly language. So, get the new CDL Linker and start writing programs in a fraction of the time it took before. Linker is compatible with Macro I & II as well as TDL/Xitan assemblers version 2.0 or later. $79.95/$20.

DEBUG I

Many programmers give up on writing in assembly language even though they know their programs would be faster and more powerful. To them assembly language seems difficult to understand and follow, as well as being a nightmare to debug. Well, not with proper tools like Debug I. With Debug I you can easily follow the flow of any Z80 or 8080 program. Trace the program one step at a time or 10 steps or whatever you like. At each step you will be able to see the instruction executed and what it did. If desired, modifications can then be made before continuing. It's all under your control. You can even skip displaying a subroutine call and up to seven breakpoints can be set during execution. Use of Debug I can pay for itself many times over by saving you valuable debugging time. $79.95/$20.

DEBUG II

This is an expanded debugger which has all of the features of Debug I plus many more. You can "trap" (i.e. trace a program until a set of register, flag, and/or memory conditions occur). Also, instructions may be entered and executed immediately. This makes it easy to learn new instructions by examining registers/memory before and after. And a RADIX function allows changing between ASCII, binary, decimal, hex, octal, signed decimal, or split octal. All these features and more add up to give you a very powerful development tool. Both Debug I and II must run on a Z80 but will debug both Z80 and 8080 code. $99.95/$20.

ZAPPLE

A Z80 executive and debug monitor. Capable of search, ASCII put and display, read and write to I/O ports, hex math, breakpoint, execute, move, fill, display, read and write in Intel or binary format tape, and more! on disk $34.95/$15.

APPLE

8080 version of Zapple $34.95/$15.

NEW! TPM now available for TRS-80 Model II!

TPM*

A NEW Z80 disk operation system! This is not CP/M*. It's better! You can still run any program which runs with CP/M* but unlike CP/M* this operating system was written specifically for the Z80* and takes full advantage of its extra powerful instruction set. In other words its not warmed over 8080 code! Available for TRS-80* (Model I or II). Tarbell, Xitan DDDC, SD Sales "VERSA-FLOPPY", North Star (SD&DD), and Digital (Micro) Systems. $79.95/$25.

SYSTEM MONITOR BOARD (SMB II)

A complete I/O board for S-100 systems. 2 serial ports, 2 parallel ports, 1200/2400 baud cassette tape interface, sockets for 2K of RAM, 3-2708/2716 EPROM's or ROM, jump on reset circuitry. Bare board $49.95/$20.

ROM FOR SMB II

2KX8 masked ROM of Zapple monitor. Includes source listing $34.95/$15.

PAYROLL (source code only)

The Osborne package. Requires C Basic 2.
5" disks $124.95 (manual not included)
8" disks $ 99.95 (manual not included)
Manual $20.00

ACCOUNTS PAYABLE/RECEIVABLE (source code only)

By Osborne, Requires C Basic 2
5" disks $124.95 (manual not included)
8" $99.95 (manual not included)
Manual $20.00

GENERAL LEDGER (source code only)

By Osborne. Requires C Basic 2
5" disks $99.95 (manual not included)
8" disks $99.95 (manual not included)
Manual $20.00

C BASIC 2

Required for Osborne software. $99.95/$20.

SYSTEM/6

TPM with utilities, Basic I interpreter, Basic E compiler, Macro I assembler, Debug I debugger, and ZEDIT text editor.
Above purchased separately costs $339.75
Special introductory offer. Only $179.75 with coupon!!

$160.

This Coupon Is Worth One Hundred And Sixty Dollars Toward The Full Price Of The SYSTEM/6 Package System/6 with this coupon is only $179.95. This is a limited time offer.

$160.00

ORDERING INFORMATION

Visa, Master Charge and C.O.D. O.K. To order call or write with the following information.

1. Name of Product (e.g. Macro I)
2. Media (e.g. 8" CP/M)
3. Price and method of payment (e.g. C.O.D.) include credit card info. if applicable.
4. Name, Address and Phone number.
5. For TPM orders only: Indicate if for TRS 80, Tarbell, Xitan DDDC, SD Sales (5¼" or 8"). ICOM (5¼" or 8"), North Star (single or double density) or Digital (Micro) Systems.
6. N.J. residents add 5% sales tax.

Manual cost applicable against price of subsequent software purchase in any item except for the Osborne software.

For information and tech queries call
609-599-2146

For phone orders ONLY call toll free
1-800-327-9191
Ext. 676
(Except Florida)

OEMS

Many CDL products are available for licensing to OEMs. Write to Carl Galletti with your requirements.

* Z80 is a trademark of Zilog
* TRS-80 is a trademark for Radio Shack
* TPM is a trademark of Computer Design Labs. It is not CP/M*
* CP/M is a trademark of Digital Research

Prices and specifications subject to change without notice.

DEALER INQUIRIES INVITED.

COMPUTER
DESIGN
LABS

342 Columbus Avenue
Trenton, N.J. 08629

NEWS & VIEWS

by Sol Libes

SOL BUSINESS SYSTEMS GROUP FORMED

A users group, called ASCII, has been formed to support SOL business systems users. Membership is $10 for which members receive four newsletters yearly, a HOT-LINE for problems and other benefits. To obtain more information or to join call or write: Jerry Brockway, Suite 308, Bayside Building, Tampa Florida, 33609; telephone (813)837-4655.

CP/NET NEXT DIGITAL RESEARCH PROJECT

With CP/M 2.0, MP/M, and PL-1 now in distribution, Digital Research is turning its attention to CP/NET. CP/NET will be a control program for microcomputer networks which allows independent processors to access I/O facilities through a network. Applications range from sharing peripherals among several slave MP/M and CP/M operating systems, to networks where the slaves provide only the computing elements which depend upon the master for disk, printer, and other I/O facilities. CP/NET will be divided into three distinct programs which operate under a master MP/M, slave MP/M or CP/M system, or take the place of the BDOS in a CP/M slave. Look for release in the late summer.

CBBS ADDITIONS

The following are some new CBBS systems that are "up and running". They should be added to the listing published previously in the Jan/Feb issue.

Long Island Computer Association...	(516)938-9043
Sacramento Computer Club	(916)483-8718

PASCAL LIBRARY IN OPERATION

Jim Gagne, DataMed Research, 1433 Roscomare Rd, Los Angeles, CA 90024 (213/472-8825) has assumed the role of interim librarian for the UCSD PASCAL User Group. He is publishing a newsletter and furnishing copies of the four volume of software currently in the library. The library disks are available for $10 + tax, each, and are furnished on 8" single-density, USCD or CP/M format. All library material may be freely copied as long as it is not sold for profit.

If you donate accepted software to the library you get a free volume of your choice. He is looking for disk editors to collect, organize, check out, document and catalog disks. Disk editors will be listed as editor of the disk, receive $1 for each disk sold and receive copies of all of the Group's disks.

S-100 COMPUTER CLUB ADDITIONS

I have received notification from several other computer clubs that either have S-100 User Groups or are exclusively S-100 oriented. They are the following:

Utah Computer Association

Has Software, hardware, CP/M and UCSD PASCAL User Groups. 378 E. 9800 So., Sandy, Utah, 84070; contact: Larry Barney (801-571-9661) or Scott Nelson (801-571-1335).

Evansville, Indiana Computer Club

Has SOL, ALTAIR and CP/M User Groups. Write:Evansville Computer Club, c/o National Sharedata Corp., POB 3895, Evansville Ind. 47737 or call: Bob Herrdink 812-426-2725.

Australian User's Group

Called "80AT" (80 Applications Transfer) has a CP/M user group and has as its primary function software exchange. It maintains an active software library. For information write: 80AT, C/- Planet 3 Systems, 47 Birch St., Bankstown, NSW 2200, Australia.

Boston Computer Society

North Star and CP/M User Groups. Write to: The Boston Computer Society, 17 Chestnut Street, Boston MA, or call: Gary Saxton (617) 816-6600 ext-2707 (days) or 877-6456 (evenings).

New England Computer Society

CP/M User Group. Write to: NECS, POB 198, Bedford MA, 01730, or call: Dave Mitton (617) 493-9362.

"C" USER GROUP NEWLETTER PUBLISHED

The first issue of the "C User Group Newsletter" has appeared in print. For the present it is available free of charge (although I personally feel that a $2-3 contribution should be sent to Acover postage, printing and misc.). Also, a disk contraining C-programs is available (disk and return postage must be provided). For more information write: C USER GROUP, P.O. Box 2556, Tallahassee Fla. 32304, or call: (904)644-2764 between 0730-0930 and 1600-2000 EST or (904)224-1101 between 1000-1530 EST.

CP/M USERS MAGAZINE PUBLISHED

LIFEBOAT ASSOCIATES, the prime distributor of CP/M system software, has started publishing a monthly magazine for CP/M users. Although to a large extent it is a promotional vehicle for Lifeboat it contains a wealth of information.

The following is quoted from the LIFEBOAT news release:

"Lifeboat Associates is pleased to announce the creation of LIFELINES, a monthly newsletter specifically designed for those who take computers and computer software seriously. The primary objective of LIFELINES is to give software owners full after-sale service by keeping readers informed on the current status of software products. Each month there will be a table listing the array of serious CP/M compatible software products distributed by Lifeboat Associates. Readers will be notified of new versions, new products, discovered bugs and bug fixes. Articles dealing with the relative merits of alternative software products will be featured.

Another objective of LIFELINES is to act as a forum for software users. Subscribers will be provided with the opportunity to give effective feedback to authors and distributors and to share experiences and concerns.

LIFELINES will also serve as the offical newletter of the CP/M Users Group (CPMUG). A section of LIFELINES will be devoted to the distribution of users group's news; catalogs and abstracts of new CPMUG volumes will be published."

The first issue contained the following goodies:

1) List of all CP/M software distributed by Lifeboat with the current release version indicated. (It was distressing to find out that every CP/M based software package I owned is out of date.) Descriptions of many of the new versions were given explaining the differences between the new and old versions.

2) Known bugs in Lifeboat distributed software were listed for many of the packages. This article listed about a dozen or so bugs. Somehow I have the feeling that the list should have been many times larger.

3) There was a delightful article by Ward Christensen titled "Users Group News". Ward is one of the most active contributors to the CP/M User Group Library and gives some good insight into the workings of the library. He also comments on some of the more worthwhile programs in the library.

4) There was a listing of three of the new disks in the CP/M user library together with abstracts of the contents of the disks. Regretfully only three out of the nine new disks were described. Hopefully this listing will continue in the next issue of the magazine.

The first issue was 17 pages and carries a price of $2.50 which appears a bit steep to me. Harris Landgarten, the editor of the magazine tells me that the following issues will be larger in size. The "introductory" price is $18 for 12 issues (USA, Canada & Mexico), $40 elsewhere. It will be mailed First Class or Air Mail. To subscribe write to: **LIFELINES**, 1651 Third Avenue, NY NY 10028.

Z USERS GROUP FORMED

A new Users Group has been formed to support Ithaca InterSystems software.....presently consisting of PASCAL/Z, Z-80 and Z-8000 software. It aims to assist users and provide a means of software exchange. A flyer will be issued bimonthly with bug notes, fixes or anything else of interest to the group. It will cost $6 to get on the mailing list. Public domain programs that run under CP/M, single sided/single density, will be distributed. The first volume is due for distribution on July 1st. There will be a charge of $10 per disk, which includes the disk and mailing. Plans call for 5 disks to released in the near future. The mailing list flyer will announce each new volume. There is no membership fee and the organization will be non-profit.

Donations will be distributed with full credit and comments. For information contact: ZUG, 7962 Center Parkway, Sacrement CA, 95823.

DIGITAL RESEARCH EXPECTS HUGE SALES INCREASE

Talk about luck. Back in 1975 Gary Kildall developed CP/M as an 8080-based disk operating system to work with Intel's PL/M development language. Gary had worked as a software consultant to Intel and helped develop PL/M. He developed CP/M on his own expecting that Intel would grab it up. But no such luck. Intel was more interested in selling hardware and did not yetappreciate the importance of software. Little did they realize that CP/M would become the standard for microcomuter operating systems and generate millions of dollars in sales.

Gary began to distribute CP/M mainly by word of mouth recommendations. Within three years sales rose to over $1 Million. Today DR employs 16 people and expects 1980 sales to hit $3 Million. They are planning to double in size and reach $30 Million next year. Currently CP/M is licensed to 200 manufacturers and software houses and there are over 400 applications packages which run under CP/M.

DR currently also offers MP/M, a multitasking DOS and PL1. Soon to be introduced is CP/NET for microcomputer networking and an operating system for 8086, 16-bit, based systems.

Now I ask you: where do you think we would be today if Intel had bought CP/M? Also, where would Gary Kildal and Digital Research be?

RUMOR

I received a phone call from an IC manufacturer who is seriously considering introducing a two-IC chip set which will handle all the interfacing requirements for the S-100 bus. This would reduce the number of ICs required on both Master and Slave cards and reduce product cost. The ICs will provide all the necessary bus buffering, control signal and address decoding and DMA logic and will meet the IEEE S-100 specs.

NORTH STAR USER GROUP FORMED

An international North Star User Association (INSUA) has started. It will provide laison, feeback and fixes for North Star users. It will also work with established local North Star user groups. It will publish a quarterly newsletter and maintain and distribute software.

Membership is $15. For more information contact: INSUA, 131, Highland Ave., Vacaville CA, 95688 or telephone: (707)448,9055.

News/Views, cont'd...

INTEL RELEASES DATA ON 32-BIT MICROPROCESSOR

Intel, the recognized leader in microprocessor development, has "leaked" advance information on three new forthcoming microprocessors. Two are upgrades of the current 8086 16-bit processor and the third is a full 32-bit microprocessor. All are expected to be introduced officially in 1981.

Also, Intel will go to a new part number system. Gone will be designations such as 8088, 8086, etc. All processor chips will have an "iAPX" prefix which stands for "Intel Advanced Processor Architecture." Thus the 8086 will now be known as the iAPX-86. Adding I/O processors will make it an iAPX-86/11 or iAPX-86/12 for one or two channels, respectively. Add a math processor and it will be known as the iAPX-86/20 or combined with I/O processors it will be an iAPX-86/21, etc.

The new 16-bit microprocessors will be known as the iAPX-186 and iAPX-286. They are upgrades of the 8086 providing up to 30 and 100% performance improvements. Both will have one gigabyte of virtual memory addressing and 16 megabytes of direct addressing.

The iAPX-186 will have three 16-bit timers, an interrupt controller, clock generator, two DMA channels, and a special communications port on chip. It is thus designed for multiprocessing applications. The iAPX-286 version will be designed for multi-user applications and will have an on-chip memory manager and two I/O channels. Both devices will come in 68-pin packages.

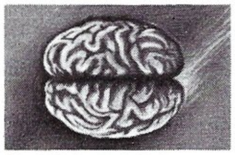
The iAPX-432 will be a 32-bit microprocessor with full 32-bit architecture. It will directly execute the ADA high level language, an extension of PASCAL. The iAPX-432 will be a 3-chip set and up to eight processors can be plugged into the bus, without changing system software, to make up one CPU with increased power.

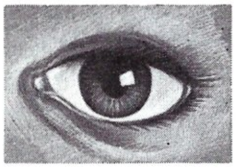
RUMORS

More rumors keep sufacing that IBM will soon introduce a new microcomputer using an IEEE S-100 bus interface. It has already been dubbed the "1505." It will be made in Japan, sell for about $4,500, include integral CRT, 16K of RAM, 30 cps printer, tape cartridge drive, ROM BASIC, and will use an 8080 type microprocessor..... Intel is expected, shortly, to announce a 60% price cut on the 8088. The 8088 is essentially a 16-bit processor with 8-bit I/O, executing the 8086 code. This reduces the 8088's price to $25 in 1,000 lot quantities and to $15 in higher quantities. It is expected that the 8088 will be under $10 within another year.

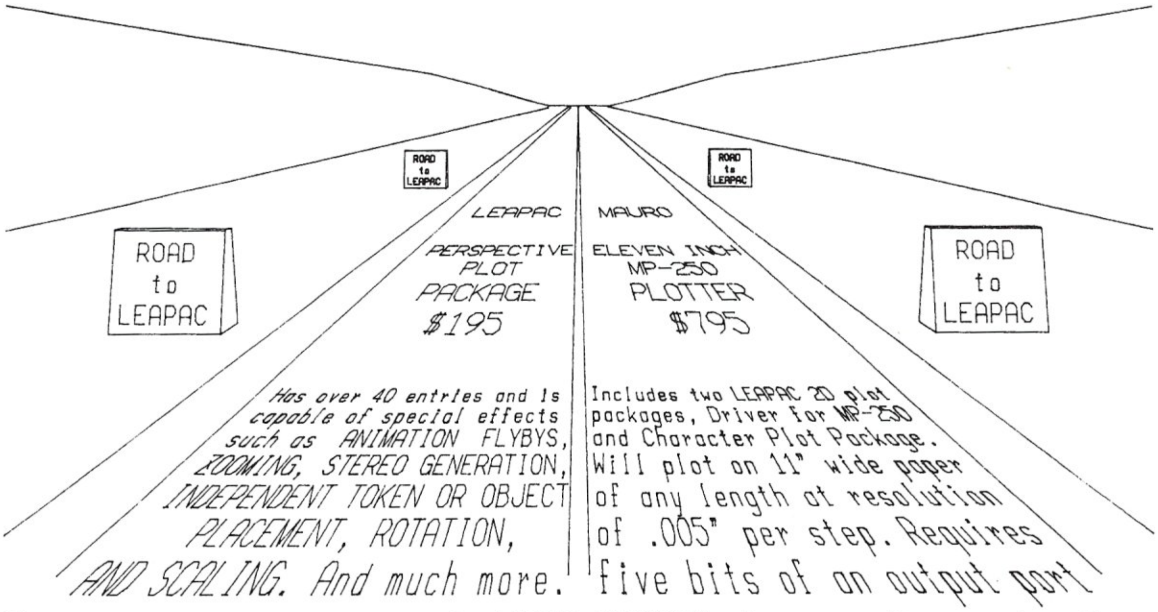
8088 CPU CARD TO BE AVAILABLE

In BYTE magazine's September and November issues there will be an article on constructing an S-100 8080 CPU card. The card was designed by Tom Cantrell, of Intel. Tom will be selling a bare CPU card with manual for $60. Also, he will make available a monitor program on disk or in ROM for $40. This could prove to be a very economical way to move up into the 16-bit CPU area. The 8088 executes the 8086 code but has 8-bit I/O. The card will work with standard 8-bit wide S-100 memory cards andI/O cards. It does lack certain IEEE features such as 16-bit request logic and pSTVAL control signal. The 2-K ROM monitor program is very powerful and includes debugging features such as multi-bread-points and single-step execution. Tom points out that Digital Research will soon release an 8088/8086 version of CP/M which should be easily adaptable to the CPU card. Further, the current 8086 microsoft assembler can be used to generate code for the 8088. For more information contact: Micro Future, PO Box 5951, San Jose CA, 95150; (408) 249-0560.





COMPLEMENTARY PLOTTING PRODUCTS



LEAPAC

PERSPECTIVE PLOT PACKAGE

$195

Has over 40 entries and is capable of special effects such as ANIMATION FLYBYS, ZOOMING, STEREO GENERATION, INDEPENDENT TOKEN OR OBJECT PLACEMENT, ROTATION, AND SCALING. And much more.

MAURO

ELEVEN INCH MP-250 PLOTTER

$795

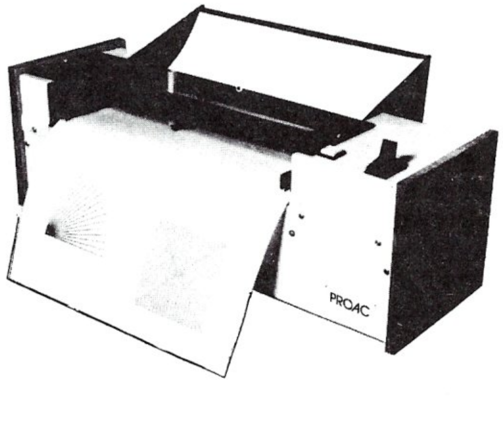
Includes two LEAPAC 2D plot packages, Driver for MP-250 and Character Plot Package. Will plot on 11" wide paper of any length at resolution of .005" per step. Requires five bits of an output port

This advertisement was created by LEAPAC SERVICES plotting software using Mauro Engineering MP-250 Plotter. The software was expressly written in ANSI FORTRAN compatable SUBROUTINE PACKAGES. They are supplied as Relative Linking Libraries on eight inch CPM soft sectored disks. The subroutines in these libraries can be selectively linked to programs compatable to MICROSOFT software products, FORTRAN-80, COBOL-80, and COMPILER BASIC.*

** CP/M is a registered trademark of Digital Research Inc.*

The above LEAPAC PERSPECTIVE PLOT PACKAGE and MAURO MP-250 PLOTTER, if purchased together, will include the perspective plot application modules, L3P-CHAR and L3P-ARC. The packages can be purchased individually. See below.

L3P	— LEAPAC Basic Perspective Plot Subroutine Package. (requires external subroutine PLOT)	$195.00
L3P-CHAR	— Perspective ASCII Character Plot Subroutine Package. (requires L3P and L2D-CHAR)	$65.00
L3P-ARC	— Perspective Elliptical Curve Plot Subroutine Package. (requires L3P)	$65.00
L2D-CHAR	— X-Y ASCII Character Plot Subroutine Package (requires ext. subroutines PLOT and WHERE)	$95.00
L2D-MPD	— X-Y Vector Plot Driver for the MAURO MP-250 Plotter. (contains PLOTS, PLOT, WHERE, FACTOR and SETORG)	$65.00
L2D-DPD	— X-Y Vector Plot Driver for 1600 type DIABLO Printers. (contains PLOTS, PLOT, WHERE, FACTOR, RESET and SETORG)	$65.00
MICRO PLOT	— Extensive X-Y Vector Plot Subroutine Package from MICAH for daisy wheel bi-directional printers such as Diablo 1620, Sprint-5, etc.	$195.00



ALL ORDERS PREPAID OR C.O.D.
TELEPHONE ORDERS ARE ACCEPTED

Prepaid — send certified or cashiers' check. Personal and company checks require 20 days to clear the banks.

C.O.D. — add $10 or 2% for handling, whichever is greater.

Deliveries in California — add 6% sales tax.

DEALER INQUIRIES ARE INVITED — Inquire about other diskette formats.

LEAPAC SERVICES
8245 Mediterranean Way
Sacramento, California 95826
(916) 381-1717

Software Product Review

THE MATE TEXT EDITOR-WORD PROCESSOR

R.D. Graham

One of the widely known benefits accruing to the S-100 computer owner is that he can use the CP/M™ operating system. One of the widely unknown benefits of using CP/M is the opportunity to use a suberb text editor-word processor called MATE. This program, designed and coded by Michael Aronson, was copyrighted early in 1979 and sold, until recently, for $69.50. Aronson, for some reason, did not aggressively market his product and it has become known to only a small sample of the CP/M users through word-of-mouth.

I have used it for six months now, writing all my assembly, FORTRAN, CBASIC, PILOT and C source programs with it and would feel crippled without it. Two years ago I had a hard time getting used to the CP/M editor (ED) but finally grew rather fond of it. However, since acquiring MATE, I have not written a program with ED. One thing more, which I'm sure will surprise Pencil enthusiasts, (and I include myself amont them); I rarely use Pencil now, finding MATE my choice in word processing, although MATE does not have all the output formatting capabilities present in PENCIL.

MATE comes with a good user manual and interface guide, and Aronson shows that he is sensitive to the documentation problem by the way he has designed and written it. The disk I received had drivers for interfacing with VDM-1, ADM-3 and Hazeltine 1500 CRT's in both HEX and ASM files. Following clear instructions in the interface guide of the manual, I had my VDM-1 version up and running without trouble.

Aronson, in the introduction to his manual, spells out the commonly accepted meanings of "text editor", "word processor" and "text output processor" and explains that "MATE is an attempt to combine some of the best features of all three". I think he has been successful in this attempt.

Mate comes up in a "Command Mode" which is reminiscent of CP/M's ED. There are a wealth of commands here, the majority of which, I must confess, I don't use much because I find it so convenient to use similar "instantaneous" commands in the "Insert Mode". In this mode, what you see is what you get. Text is entered by simply typing. Editing changes show up instantaneously on the screen at the cursor position. No more blind editing! You can move the cursor to the beginning or end of the text buffer with control A and control Z respectively, and besides moving the cursor up or down one line at a time you can move it up or down 6 lines at a time. This allows you to move through your text very rapidly. Similarly you can move the cursor forward or backward one character at a time, or one word at a time. Insertion or deletion of text at the cursor is similarly easy, instantaneous and always with the sure knowledge that it has been done correctly since you see it happen. Big blocks can be moved either with tags or with easy moves of text to one of ten text buffers available, from which it is inserted at the cursor position with another simple command.

Search, search and change, set tab stops, delete tab stops, set left and right margins are all commands (with many options) available to the user of MATE.

Users with big complicated editing jobs will probably find the macro facilities available in MATE very much to their liking for they can, in effect, add their own commands to MATE's command set. To aid in "programming" these complex macro command strings, Mate includes a breakpoint and trace facility. I have not attempted to build any macro command strings because for the uses I make of MATE I find it quite powerful enough the way it is. However, many will probably want to improve its output formatting capability and this would be one way to do so.

In summary, I think that for the money MATE cannot be beat; and that many of you will agree with me that in preparing source code files under CP/M it cannot be beat at any price.

R.D. Graham, 550 Starboard Drive, Naples, FL 33940

SO, WHAT ARE YOU WAITING FOR?

VECTOR GRAPHIC (20% off)
(all are assembled & tested)

ZCB Z80A SBC	$345.00
Z80A CPU	$195.00
Flashwriter 2 video	$295.00
Bitstreamer 2 I/O	$225.00
Precision Analog	$350.00
Hi-res. Graphics	$225.00
Video Digitizer	$160.00
18-slot mother	$160.00
PROM/RAM 3 (12K)	$195.00
8K Static RAM	$175.00
64K Dyn. RAM	$795.00
Micropolis Disk Cont.	$425.00
Vector Graphic MZ	$3495.00
System B	$4495.00
System B/Q (w/Qume)	$7395.00
NEW System 2800	$5995.00
NEW System 2800/Q	$8895.00
NEW System 3030	$AVE

OTHER SPECIALS

Zenith Z89 (48K)	$2595.00
PASCAL Microengine	CALL
CF&A desks 30" X 48"	$159.00
CF&A Univ. print. std.	115.00
(the CF&A pair) only	$259.00
Konan "Hard Tape" subsystems from	$5995.00

"THE DAISY PATCH"

Plessey "Alphagraph" printer (daisywheel w/ bi-directional tractors) SPECIAL	$2495.00
C. Itoh "Starwriter"	$1895.00
NEC & Qume printers	CALL
Anadex 9500	$1495.00
Centronics 737	$895.00
TVI 912 & 920 B&C	CALL
ADDS 25's & 40's	CALL
Microterm 2A, 5A, 100	CALL
Graham-Dorian Software	$AVE

CARDFILE is an enhanced version of Vector Graphic's Word Mgmnt. merged mailing list...more like a DBMS and it handles 9-digit zip codes! ONLY $125.00
(with MEMORITE only $500.00

25% down - balance COD
prepay saves freight
(add 4%tax on MN sales)
local prices slightly higher

~Budget InfoSystems~
1633 NE Highway Ten ~ suite 5 west
Spring Lake Park, MN 55432
786-5545 (757-5878 afterhours)

A 16-BIT WIDE MEMORY FOR THE S-100 BUS

Ken Maun & Pat Stakem

This article discusses the design of two 16 bit wide memory boards compatible with the proposed IEEE S-100 bus standard (IEEE-696). These units are compatible with the addressing modes of current 16 bit and extended 24 bit addressing. By using a piggyback card in one of the designs, different memory configurations (ROM, dynamic RAM, static RAM) may share the same controller and power circuitry. Virtual address paging is also discussed.

The IEEE proposed standard for the S-100 bus (reference 1) expands its capabilities from 8 bits of data and 16 of address to 16 bits of data and 24 of address. These features are required by the new generation of 16 bit processors such as the 9900, 8086, Z-8000, and M68000. Although 16 bit processors have been put on the S-100 bus before, these operations involved compromises such as dual S-100 busses for parallel data access, or sequential byte accesses. In addition, processors on the S-100 bus had to emulate 8080 timing and control, since the bus was essentially an 8080 bus. These restrictions have been greatly relaxed by the new bus standard, and the IEEE group is to be greatly lauded for achieving a semblance of order in a difficult area.

To support a new family of 16 bit procesors on the S-100 bus, a new series of peripheral cards are called for. This paper describes a memory card philosophy for the new S-100 standard that is compatible with both 8 and 16 bit accesses, and both byte and word mode 16 bit operations.

First, the extension of the address bus to 24 bits, and the data bus to 16 bits will be briefly discussed.

Discussion of the 24 bit addressing mode

The address bus can be viewed as 16 or 24 parallel lines. At least 16 lines are employed, with more lines available if extended addressing is desired. These lines are designated A0 (LSB) thru A15 or A23 (MSB). I/O device addresses appear on address lines A0 through A7, or A0 through A15 if extended I/O addressing is to be employed.

Discussion of 16 bit data mode

For byte operations, the "old" S-100 standard of two unidirectional 8 bit data busses is used. Data input (with respect to the bus master) appears on the DI bus and data output on the DO bus. For sixteen bit data transfers, the DI and DO busses are used together as a single 16 bit bidirectional bus, under control of a 16 bit transfer request line and acknowledge line. The acknowledge line is incorporated to allow use of current 8 bit memory and I/O boards intermixed in a system with 16 bit boards. It should be noted that in the present proposed IEEE configuration it is possible to parallel two present design memory cards, or one of present design and a new eight bit card, to accomplish alternately an 8 and 16 bit wide memory, but that this is not simple nor completely straightforward. A minor modification of the presently proposed standard would allow this. However, in this discussion such a modification will not be considered. Rather, we present two new memory card designs of flexibility and expanded capability, and a low cost alternative.

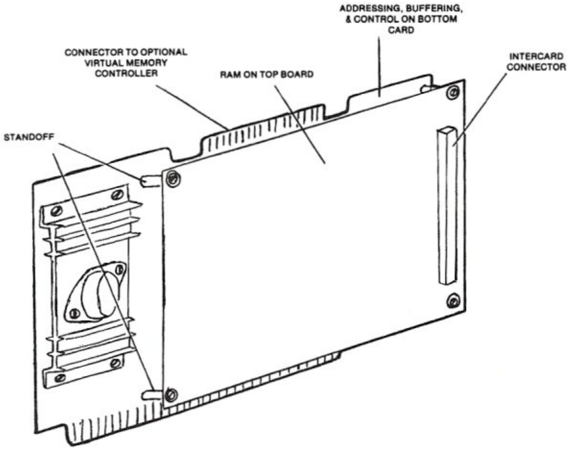

ADDRESSING, BUFFERING, & CONTROL ON BOTTOM CARD
CONNECTOR TO OPTIONAL VIRTUAL MEMORY CONTROLLER
RAM ON TOP BOARD
INTERCARD CONNECTOR
STANDOFF

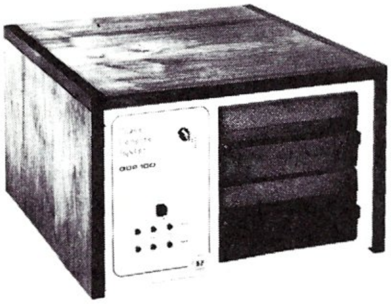
FIGURE 1

A High Flexibility Memory Unit Design

One of the main concerns in designing a byte-organized 16-bit memory is that of byte boundaries. For example, consider a CPU access of a single byte at location thirty (0011110). Next, assume that CPU tries to fetch a double byte instruction from the following location (thirty one). This would entail fetching a byte from locations thirty one (0011111) and thirty two (0100000). The memory must know to take from its

Ken Maun & Pat Stakem, Interface Technology, Box 745, College Park, MD 20740

THE NEXT GENERATION OF MICROCOMPUTERS IS HERE AT QUASAR DATA PRODUCTS



16 BIT POWER

Z-8000

AND STILL RUN YOUR 8 BIT SOFTWARE

Z-8000 SERIES 16 BIT CPU S-100 BOARD — CAN BE PLUGGED INTO YOUR EXISTING SYSTEM

- FULLY S-100 IEEE COMPATIBLE
- SUPPORTS EXISTING 8 BIT MEMORY AND 8 BIT PERIPHERAL BOARDS
- CAPABLE OF READING AND/OR WRITING 8 BIT, 16 BIT OR MIXED 8 BIT AND 16 BIT MEMORIES AUTOMATICALLY
- 8 BIT AND/OR 16 BIT PERIPHERAL MODULES CAN SIMULTANEOUSLY CO-EXIST IN THE SAME BUS WITHOUT ANY MODIFICATIONS
- CAPABLE OF OPERATING AS A SLAVE PROCESSOR TO ENABLE YOUR EXISTING CPU TO CONTROL THE Z-8000
- SUPPORTS ON-BOARD HARDWARE SINGLE STEPPING
- SUPPORTS EITHER SEGMENTED CPU OR NON-SEGMENTED CPU
- POWER-ON AND RESET JUMP DIP SWITCH SELECTABLE
- JUMPER SELECTABLE 2 OR 4 MHz. OPERATION
- DIP SWITCH SELECTABLE NUMBER AND TYPE OF WAIT STATES
- SOFTWARE
  - Z-80 EMULATOR ENABLES YOU TO EXECUTE YOUR EXISTING 8 BIT SOFTWARE **WITHOUT** ANY MODIFICATIONS AND ALLOWS YOU TO RUN CP/M IMMEDIATELY
  - EXTENDED MONITOR, DEBUGGER, DISASSEMBLER

INDUSTRIAL QUALITY

Z-80

Z-80 SERIES 8 BIT CPU S-100 BOARD

- 4 MHZ. Z-80 MICROPROCESSOR
- FLOPPY DISK CONTROLLER ALLOWING SINGLE- AND DOUBLE-DENSITY USING EITHER 8" or 5¼" DISK DRIVES
- ROOM FOR THREE 2716 EPROMs or THREE 2316 ROMs
- TWO RS-232C SERIAL PORTS
- TWO PARALLEL PORTS
- HARD DISK CONTROL VIA THE PARALLEL PORTS
- REAL TIME CLOCK
- RESET-JUMP CIRCUIT ALLOWS CPU TO JUMP TO MONITOR SOFTWARE ON RESET
- CAPABLE OF PROGRAMMING AND VERIFYING INTEL 2716 ON BOARD WITH EXTERNAL POWER SUPPLY
- UTILIZES VECTORED INTERRUPTS OF Z-80 CPU
- MONITOR PROVIDED

INDUSTRIAL QUALITY

SYSTEMS

QDP-8100 WITH 2 MEGABYTES STORAGE STANDARD (OPTIONAL 4 MEGABYTES)

- Z-8000 SERIES 16 BIT CPU S-100 BOARD - SEE ABOVE
- SOFTWARE (PROVIDED WITH SYSTEM)
  - CP/M 2.2[1] OPERATING SYSTEM
  - BASIC
  - Z80/8080 EMULATOR
  - MONITOR, DEBUGGER, DISASSEMBLER
  SOFTWARE OPTIONS: PASCAL
  - UNIX[2] OPERATING SYSTEM COMING

QDP-100 WITH 2 MEGABYTES STORAGE STANDARD (OPTIONAL 4 MEGABYTES)

- Z-80 SERIES 8 BIT CPU S-100 BOARD - SEE ABOVE
- SOFTWARE (PROVIDED WITH SYSTEM)
  - CP/M 2.2[1] OPERATING SYSTEM
  - BASIC
  - ACCOUNTS RECEIVABLE, GENERAL LEDGER, ACCOUNTS PAYABLE, PAYROLL WITH COST ACCOUNTING
  - OPTIONAL SOFTWARE: FORTRAN, PASCAL, COBOL, C

EACH SYSTEM CONTAINS:

- INTELLIGENT CRT TERMINAL (80 CHARACTERS X 24 LINES)
- 64 KBYTES RAM
- TWO 8 INCH, DOUBLE SIDED, DOUBLE DENSITY FLOPPY DISK DRIVES WITH CONTROLLER
- 2 SERIAL AND 1 PARALLEL (2 PARALLEL FOR QDP-100) PORTS
- ATTRACTIVE WOODGRAIN CABINET WITH POWER SUPPLIES AND CABLING

FULL TECHNICAL SUPPORT FROM THE STAFF AT QUASAR DATA PRODUCTS

[1] CP/M™ DIGITAL RESEARCH
[2] UNIX™ BELL LABS

4 Mhz 64K Dynamic RAM

16K - $^{\$}$**250**^{00} 32K - $^{\$}$**350**^{00} 48K - $^{\$}$**450**^{00} 64K - $^{\$}$**549**^{00}

TELETEK DBL. DENSITY, DBL. SIDED

Disk Controller Board.............................. $^{\$}$**395**^{00}

QUASAR FLOPPY SYSTEM

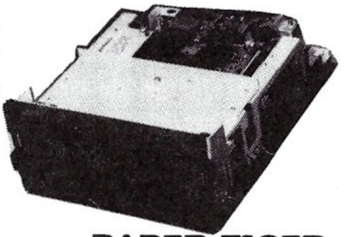
- Two MFE DBL sided drives • Cable • Case & Power Supply assembled and tested Wood cabinet .... $^{\$}$**1895**^{00}

QUASAR 2 MEG FLOPPY

- 2 MFE double sided drives
- Teletek disk controller board
- Power supply & cable
- Wood cabinet
- CP/M version 2.2 & bios
- Assembled & tested .... $^{\$}$**2295**^{00}

Dealer Inquiries Invited, Hours:9-5:30 M-F

Specifications Subject To Change



MFE Double Sided - Double Density 8" Floppy Disk Drives. (the best) ..... $^{\$}$**650**^{00}
Using the Teletek Controller under CP/M, THIS DRIVE WILL GIVE YOU ALMOST ONE MEGABYTE PER DISK DRIVE.
Power supply for above .... $^{\$}$**110**^{00}

PAPER TIGER

Includes Graphics .... $^{\$}$**949**^{00}
Cable for TRS-80 ...... $^{\$}$**39**^{00}
Call for Apple

TI - 820

Serial Printer -
Full package options... $^{\$}$**1995**^{00}

UnixTM - Bell Lab 30 Day ARO CP/MTM - Digital Research

Checks, money orders accepted
Add $2.50 freight charges on orders under 10 lbs. Over 10 lbs. F.O.B. Cleveland

QUASAR DATA PRODUCTS

25151 Mitchell Dr., No.Olmsted, Ohio 44070 (216)779-9387

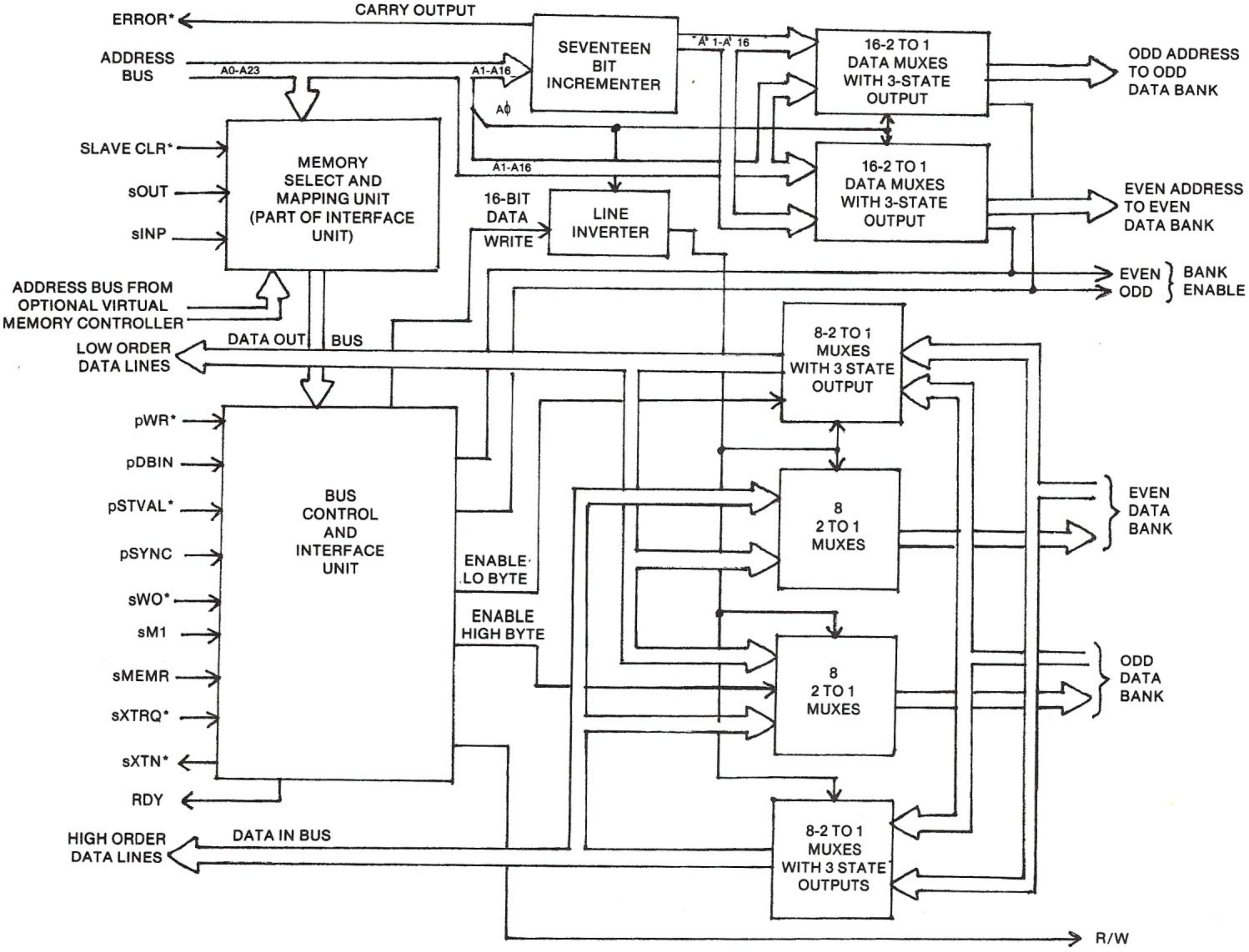
16-Bit, cont'd...

odd-byte memory and put data onto the lower lines, and to take from its even-byte memory and put data onto the upper lines. (At other times, it would have to put the data out in reverse order of this.) Additionally, in the case above, in which the address is given in the odd byte, the memory would have to know to access a completely different address for the even byte. In the above case, it is 001111X for the odd byte, and 010000X for the even byte. This is the age old problem faced by mini-computer designers. Some have ignored the problem, only to be faced with programming problems in the field. It is a serious consideration.

For some of the 16-bit micros, such as the 9900 which addresses only on even-byte boundaries this is no problem (the 9900 doesn't have an A0 line). Others, such as the 8086, addressing can be on either even or odd byte boundaries. To force this machine to observe only even boundaries, would pose considerable hardship on the programmer; however, its designers anticipated the problem and have constrained the 8086 to access odd boundary words in two steps, a byte at a time. It seems no current 16 bit processor addresses a word on odd (byte) boundaries. This simplifies the design.

The design series in figures one through four can operate on either odd or even boundaries. It can also transfer data as either a 16-bit word or an 8-bit byte on either boundary. It can be implemented as either a static or a dynamic system. The dynamic implementation could have more features -- some of whose cost would not be justified on the lower capacity static units. The family would justify a 16K byte static version, a 32K byte static version, and a 128K byte dynamic version. The number of features would be directly proportional to the capacity. Since the others are only a subset of the dynamic unit, only it will be discussed here.

The basic unit is shown in figure 1. Since the unit, of necessity, has a good deal of support circuitry, a double deck S-100 card is indicated. The easily disconnected top card would contain the actual RAM. In case of the dynamics, this would have a capability of sixty-four chips, in a configuration very similar to the present 8K byte static units that proliferate on the market. The double-depth card allows compatibility with existing cabinets, and the thermal considerations



ERROR*
CARRY OUTPUT
ADDRESS BUS
A0-A23
A1-A16
SEVENTEEN BIT INCREMENTER
A' 1-A' 16
Aϕ
16-2 TO 1 DATA MUXES WITH 3-STATE OUTPUT
ODD ADDRESS TO ODD DATA BANK
SLAVE CLR*
sOUT
sINP
MEMORY SELECT AND MAPPING UNIT (PART OF INTERFACE UNIT)
A1-A16
16-BIT DATA
WRITE
LINE INVERTER
16-2 TO 1 DATA MUXES WITH 3-STATE OUTPUT
EVEN ADDRESS TO EVEN DATA BANK
EVEN
ODD
BANK ENABLE
ADDRESS BUS FROM OPTIONAL VIRTUAL MEMORY CONTROLLER
LOW ORDER DATA LINES
DATA OUT
BUS
8-2 TO 1 MUXES WITH 3 STATE OUTPUT
pWR*
pDBIN
pSTVAL*
pSYNC
sWO*
sM1
sMEMR
sXTRQ*
sXTN*
RDY
BUS CONTROL AND INTERFACE UNIT
ENABLE LO BYTE
ENABLE HIGH BYTE
8 2 TO 1 MUXES
EVEN DATA BANK
8 2 TO 1 MUXES
ODD DATA BANK
HIGH ORDER DATA LINES
DATA IN BUS
8-2 TO 1 MUXES WITH 3 STATE OUTPUTS
R/W

FIGURE 2

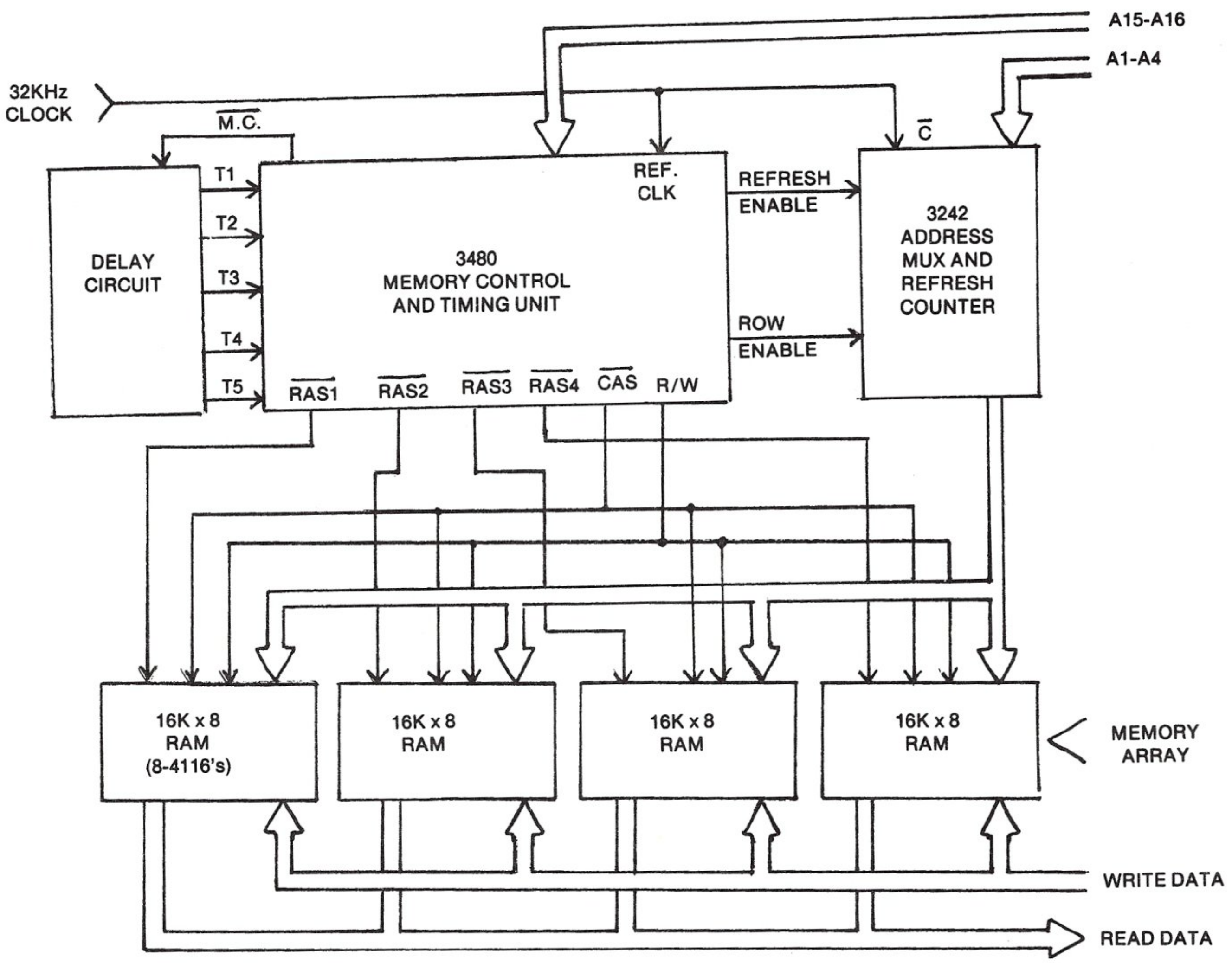


FIGURE 3

have been covered by keeping the large heat dissipating elements (memory chips and the power regulators) completely exposed to the air. Figure 2 shows the basic control and interface circuitry, including the incrementer which allows us to address words on both odd and even byte boundaries, the data bus multiplexers which control the data traffic flow, and the memory interfacing, select, mapping and control units which allow the memory to be dynamically configured and mapped throughout the address space.

Figure 3 depicts one of the two 64K x 8 bit memory banks.

ADDRESSING

In order to achieve the byte-boundary addressing discussed above, the incrementer, shown in figures 2 and 4 is used. This is essentially an adder, which always adds "one" to any address. The unincremented address is supplied to the bank specified on the address boundary. Thus, if an odd-byte address is received, whatever this address is will be sent to the odd-byte memory bank (minus A0). The incremented value would be sent to the even-byte memory bank (also minus A0). A possible incrementer configuration is shown in figure 4. Simpler configurations are pos-

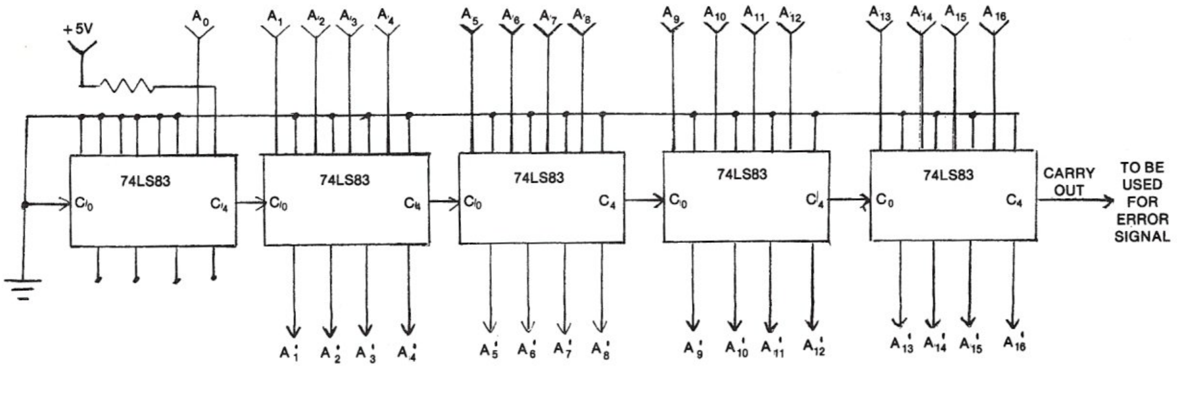


FIGURE 4

16-Bit, cont'd...

sible. In the more inclusive implementations of this system, each 1K byte portion of each memory could be assigned to any location within the entire 16 megabyte memory space, thus allowing complete virtual memory management and multi-program operation. In addition, read protect and write protect could be dynamically controlled. A Virtual Memory controller card to accomplish this will not be discussed here.

OTHER CONSIDERATIONS

1. Top Byte of Memory: One difficulty in this arrangement is in word-addressing the very top of a memory card (byte FFFF). When this is done the lower byte accesses the top of memory, and the upper byte falls off the edge. Even when another memory card follows this one, it has no way of knowing it has been addressed. (Actualy the addressing wraps around to the bottom of the card, but this is not what the programmer usually wants. Rather than increase the cost, just to get one-byte of data, a top of memory error line is provided to alert the programmer not to access this word. (It only occurs once in 128K bytes).

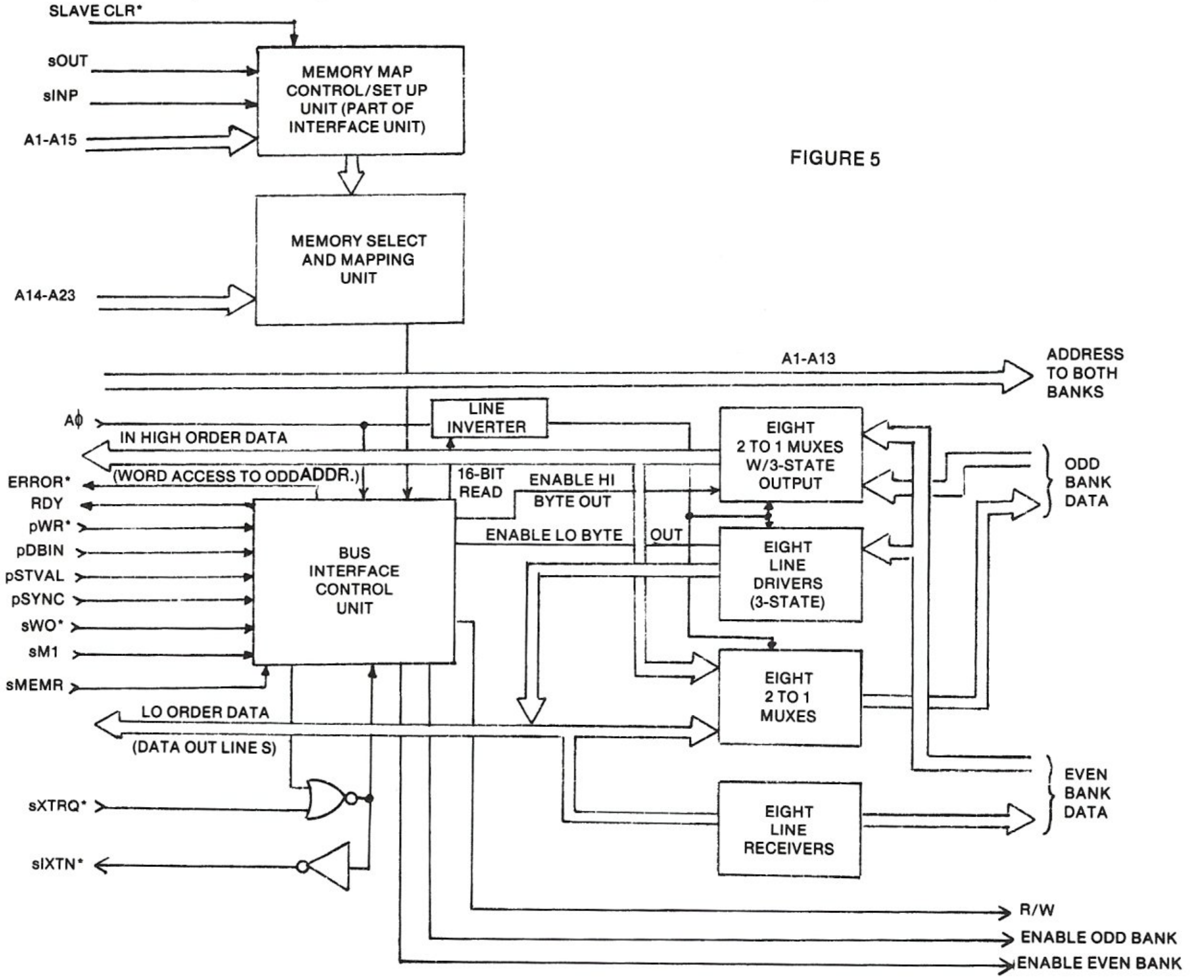


FIGURE 5

2. Parts Count: The number of auxiliary chips used is relatively high; however, the cost per bit is quite low.

3. Expandability: This design is capable of later expansion to a half megabyte capacity on a single card using the 64K bit dynamic RAM. As discussed earlier, the cards will be compatible with virtual memory mapping.

4. Power: Regulated power is provided by the controller card, and is derived from the S-100 bus unregulated lines.

5. As mentioned previously, the popular present 16-bit micros inhibit odd-byte word addressing; however, there is no guarantee that this will always be the case. With this memory, the user is protected in any eventuality, be it a new and different micro, a bit-slice design or a 16-bit controller.

A SIMPLE, LOW COST MEMORY UNIT DESIGN

For those who do not need all the bells and whistles of the design just presented, a simpler implementation is suggested. It doesn't have odd-byte word addressing capability or virtual memory or multiprogramming facilitation capability.

This design, illustrated in figure 5 is simply intended to provide the capability for a reliable, relatively low-cost memory, usable by both 8 and 16 bit masters.

Figure 6 through 11 show the operation of this design in different access situations. The first of these (figure 6), for example, shows the flow of data in the case in which an 8-bit master (CPU) is writing to an even address. Here, data is received by the memory unit via the low order data (OUT) lines and presented via the proper line receivers to the "EVEN" memory data bank. At the same time, this (EVEN) bank is enabled to receive data.

Figure 7 shows an 8-bit write to an **odd** address. Figure 8 depicts an 8-bit read from an **even** address. Note that, here, the High Order (IN) data lines are used.

derives its data. Figure 9 shows an 8-bit read from an odd address.

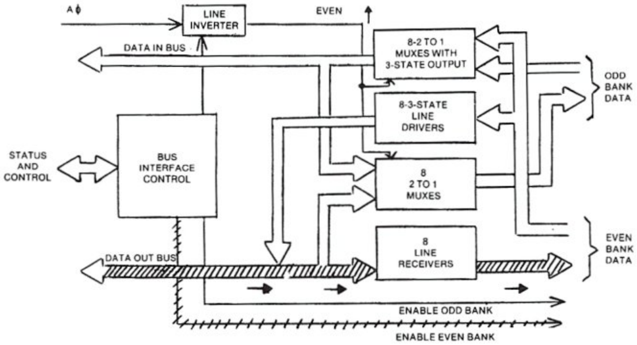


FIGURE 6

Also note that the A0 signal is used as a "steering" signal to determine from which bank the output MUX

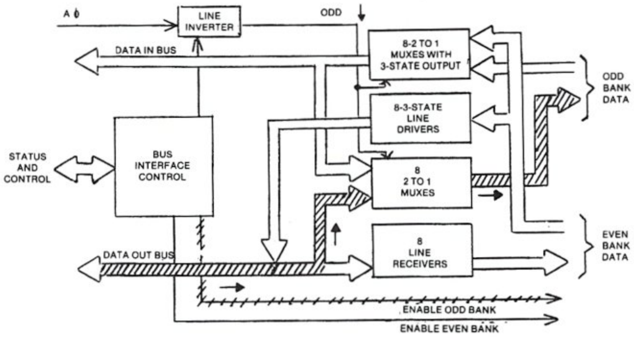


FIGURE 7

The remaining two figures are used to depict sixteen bit transfers. It must be remembered that, in this simpler configuration, word transfers are not allowed on odd byte boundaries, thus there are only two possibilities, write and read. Figure 10 illustrates a 16-bit write operation and figure 11 a 16-bit read operation. In both cases, both memory banks are enabled. Note that for a 16-bit read (figure 11) a special provision must be automatically made. In other cases, when A0 was even it automatically steered the output muxes to the proper banks for even data. When odd, A0 steered the muxes to proper banks for odd data; not in this case. Due to an idiosyncrasy of the way the IEEE S-100 BUS spec is now set up, a special line inverter must be included to invert the convention in this case. This inverter can be implemented with a single exclusive-or gate.

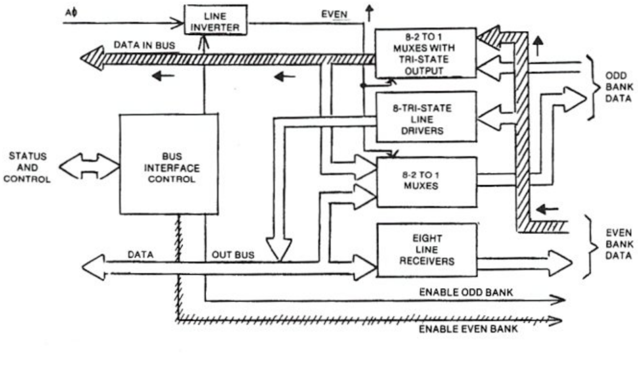


FIGURE 8

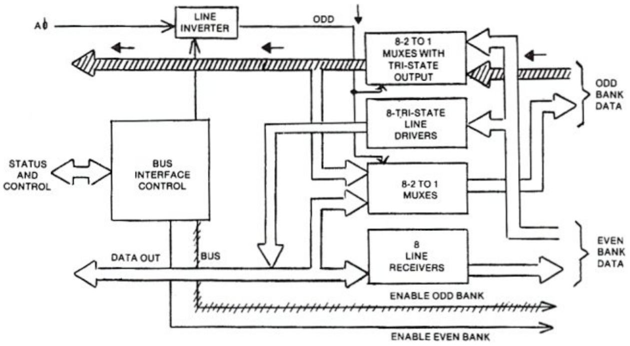


FIGURE 9

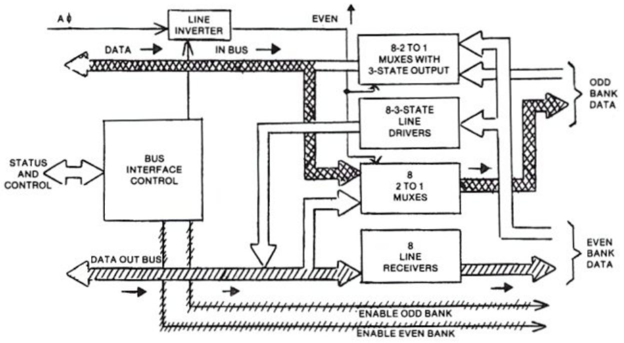


FIGURE 10

Unlike the previously discussed, high flexibility design, this simpler configuration can be implemented on a single card. As envisioned, it would also probably be best limited to simple static ram implementations of 8 to 16K byte capacities.

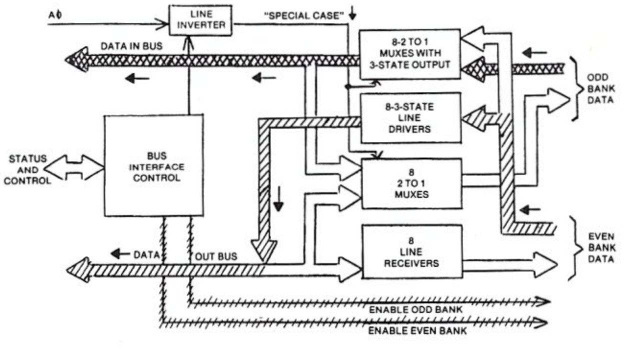


FIGURE 11

REFERENCES

1. "Standard Specification for S-100 Bus Interface Devices", Elmquist, et. al, IEEE Computer, July 1979.
2. "Proposed Standard for the S-100 Bus", Morrow and Fullmer, IEEE Computer, May 1978.
3. "A Designers Notes on the S-100 Bus Standard Proposal", Elmquist, Interface Age, August 1978.
4. "Introducing the S100: Standard Small Computer Bus Structure", Goble, Interface Age, June 1977.

8080 DYNATRACE

Charlie Foster & Richard Meador

A super 8080 emulator program useful in debugging 8080 programs.

Anyone who is learning to program in assembly language can use a method of observing just what is going on inside of the CPU. If you know how to program already, you still need a way to debug your new programs. Dynatrace will help you in either case. Dynatrace is a development tool that will accept commands from any standard ASCII keyboard and provides a TWO PART DISPLAY. The video monitor is configured for a 64 character by 16 line display but since this article includes the source any other configuration can be patched into the program.

The upper 4 lines of the display are dedicated to a dynamic display of the contents of register information being used in conjunction with the program being developed. The register display is always in view and is updated continuously as simulation progresses whether the simulation is single step or continuous run. Dynatrace is actually a pseudo-computer simulating in software everything done by the 8080 in hardware and more. With Dynatrace the user is able to see at a glance all register information and can make changes to existing contents of registers as desired.

Dynatrace is also easily reconfigured to utilize subroutines existing in a monitor the user may already have up and running. The A, E, F, H, J, K, N, O, P, Q, T, U, V, W, X and Y commands, being undefined, provide the user with the facilities for extensive expansion of the Dynatrace command set. Unused commands may be implemented by storing the address of a subroutine in the jump table beginning at address Base + 4AH. Thus, if the base of your Dynatrace is 0400H and you have a subroutine which you wish to incorporate at address 0324H and thereby define the A command, you need only store 24H at address 044AH and 03 at address 044BH. Once the new command has been defined, the user need only type the capital letter corresponding to the command he has just defined to call his subroutine from dynatrace. Any subroutine the user may already have in ROM may be incorporated as a command by the above method, or if used only infrequently it may be called and executed with the C command. If the user so desires, he may expand the size of Dynatrace by entering subroutines beginning at address Base + 0C00 hex. In its present form it takes up about 3K of memory. As you can see, the source is so heavily documented that it is close to 54K.

Commands

Commands are given to Dynatrace by typing a single capital letter followed by amplifying data as described below. All addresses and values are given in hex and all hyphens are issued by Dynatrace as prompting characters. Carriage returns for indicating the end of an entry are not required if the value being entered is of the length expected. Thus, typing 4 hex digits for an address value of 2 hex digits for a byte value will complete the entry and not require a terminating character. Any value being entered with fewer digits than required will, however, require a terminating character such as a carriage return.

B	Toggle binary display of scratch registers and accumulator
C-xxxx	Call the user subroutine at address xxxx. The displayed register data is loaded into the 8080 hardware registers and a normal subroutine call is made to the above address. A return to Dynatrace is effected by maintaining proper stack discipline and executing a normal subroutine return instruction. Upon return, the contents of the hardware registers are stored in the user's registers and displayed on the screen of the monitor. The C command must have the user's stack pointer defined to be in some area of existing RAM not used by Dyna-

Charlie Foster, The Brain-Drain Co., 7962 Center Parkway, Sacramento, CA 95823

trace. This is set up initially for the user, so no problems should occur. However, if the user inadvertantly changes the stack pointer to some area in the address space where RAM does not exist, or where it interferes with a stored program (either under test or with Dynatrace itself), results will be unpredictable.

D
FROM-xxxx
TO-yyyy

Display the contents of memory from address xxxx to yyyy. The D command does not like addresses with unlike signs.

G-xxxx
STOP AT-yyyy

Causes simulated execution of a user's program to begin at address xxxx and continue until address yyyy is encountered or any key is typed on the keyboard. The speed of simulation is selected with the "I" command below.

Ix

Sets instruction execution speed from 0 to F hex where 0 is approximately 1 instruction per second and F is about 200 instructions per second.

Lrxxxx

Load register pair with xxxx where "r" is the first letter of the register pair name. (LH1234 put 1234 in the HL register pair.) Note: The accumulator and the Processor Status Word are concatenated to form a register pair "APSW."

Mxxxx-yy-

Store yy at memory location xxxx and prompt with a hyphen for more data. Each succeeding byte will be placed in a subsequent memory location. The entry of data continues until a carriage return is entered in place of information. The M command has no provision for backspacing to delete an erroneously entered character. In the event that an entry is made incorrectly, it will be necessary to terminate the current command line and begin entering data after the last correct entry. The data entered before the error entry will have been stored correctly in memory and hence need not be repeated.

R

Read Intel format papertape from teletype or reader.

S

Causes the simulated execution of one instruction at the location in memory indicated by the user's program counter.

Z
FROM-xxxx
TO-yyyy

Zero RAM from address xxxx up to but not including yyyy.

Peculiarities

1. Due to the nature of the execution of certain instructions, the ending of the execution of one instruction and beginning of the execution of the subsequent instruction does not always correspond with the display. The simulation, however, is always carried out correctly and causes no problems in the program under development. The inconsistency is the updating of the program counter which is delayed one instruction for jumps, calls and returns.
2. Dynatrace is not ROMABLE as is, but if the user has a need for Dynatrace in ROM he can contact the authors to make arrangements to customize Dynatrace to his system. If there is any other need for customizing, the authors are willing to discuss the problem. Just send a self-addressed,
3. Typing C-xxxx, where xxxx is the base address of Dynatrace, can be used to restart Dynatrace and clear the screen of any previously entered data. The contents of memory are not disturbed, providing a convenient way to clear the user's registers and reset the stack pointer to its default value (C-8000 in this version).
4. If the user so desires, he may expand the size of Dynatrace by entering subroutines beginning at address Base + 0C00 hex.
5. Your system must be memory-mapped.
6. All commands must be in capitals. An "Escape" will abort an entry.
7. Displays will be at top of screen for registers. The lower middle is for memory read out and the bottom for command lines.

Further Notes

I would like to say something about how to get this program up and running. First of all, it must be edited for the change in EQUATEs that will allow it to run on the user's system. (In my case, I only needed to change the Video and Keyboard equates.) Then it must be assembled. Now, the user only needs to use DDT to call up DYNATRACE.HEX. Once there, type G8000 and DDT will jump into DYNATRACE. To return to DDT send DYNATRACE to a memory location with a RST 7. (If you don't know where one is, use Dynatrace to write one into memory. Then use the C command to go there.)

Dynatrace, cont'd...

When you want to debug a program you only need to use DDTs' "I" command to call it up and the "R" command to read it into memory. From there you are on your own. In my system I have a 4K monitor residing in EPROM, so not only do I use Dynatrace—I use my built-in monitor subroutines, too.

If you prefer a COM file, use a Relocatable assembler such as Microsofts' M80 or Cromemcos' ASMB. They can place Dynatrace at any location that you would want. A COM file would have to be placed at 100H.

Conclusion

Finally, as you can see, the program is a long one to type. So for those who would prefer to have the source already on a disk, the authors can provide a copying service for a limited period of time (until Jan. 1, 1982). If the reader will send a self-addressed, stamped shipping package with a disk, the authors will copy and mail their package (by return mail) for a handling fee of $5.00. For those who don't want to bother with any of that, just send $25.00 and the authors will provide everything. At least until the price of materials goes up. Note: The disk will be CPM/soft-sectored/single density.

The authors can be reached at:

THE BRAIN-DRAIN CO.
7962 Center Pkwy
Sacramento, CA 95823

THE DYNATRACE SOURCE

DYNATRACE
V 2.0

Copyright © 1976, by Richard E. Meador, Sacramento, CA 95826
All rights reserved with the exception of reproduction by those individuals for their own noncommercial use.

Edited for publication by Charlie Foster, 1980.

```
         ORG     08000H
DYNAT:   DS      0       ;BASE ADDRESS OF PROGRAM
STACK    EQU     DYNAT+0C00H     ;SET UP SYSTEM STACK
CONPRT   EQU     0C8H    ;CONTROL PORT ADDRESS
SCREEN   EQU     0CC00H  ;1K OF BUFFER FOR SCREEN
TOS      EQU     (SCREEN+1024)./256     ;LAST BUFFER ADDRESS+1
MIDSCR   EQU     SCREEN+448      ;TOP LINE OF ROLLUP PORTION OF SCREEN
LINE     EQU     64      ;64 CHARACTERS/LINE
LINE1    EQU     SCREEN+LINE
LINE2    EQU     LINE1+64
LINE3    EQU     LINE2+64
LINE4    EQU     LINE3+64
LINE5    EQU     LINE4+64
LINE15   EQU     SCREEN+960
PCH      EQU     LINE1+8 ;PC HEX DISPLAY LOCATION
INSTA    EQU     LINE2+8 ;INSTRUCTION ASCII DISPLAY LOCATION
INSTH    EQU     LINE2+18        ;INSTRUCTION HEX DISPLAY LOCATION
SB       EQU     LINE4+8 ;SIGN FLAG BINARY DISPLAY LOCATION
ZB       EQU     LINE4+9 ;ZERO FLAG BINARY DISPLAY LOCATION
ACB      EQU     LINE4+11        ;AUX CARRY FLAG BINARY DISPLAY LOCATION
PB       EQU     LINE4+13        ;PARITY FLAG BINARY DISPLAY LOCATION
CYB      EQU     LINE4+15        ;CARRY FLAG BINARY DISPLAY LOCATION
ACCH     EQU     LINE1+22        ;ACCUMULATOR HEX DISPLAY LOCATION
ACCB     EQU     LINE1+25        ;ACCUMULATOR BINARY DISPLAY LOCATION
BCH      EQU     LINE1+37        ;B REG HEX DISPLAY LOCATION
BCB      EQU     LINE1+42        ;B REG BINARY DISPLAY LOCATION
DEH      EQU     LINE2+37        ;DE REG PR HEX DISPLAY LOCATION
DEB      EQU     LINE2+42        ;DE REG PR BINARY DISPLAY LOCATION
HLH      EQU     LINE3+37        ;HL REG PR HEX DISPLAY LOCATION
HLB      EQU     LINE3+42        ;HL REG PR BINARY DISPLAY LOCATION
SPH      EQU     LINE4+37        ;SP REG PR HEX DISPLAY LOCATION
KSTAT  EQU     6EH
KEYBRD   EQU     6CH
KBDRDY   EQU     80H
RSTAT    EQU     07H
READER   EQU     06H
RDRRDY   EQU     02H
SWTCHS   EQU     0FFH
CR       EQU     0DH
LF       EQU     0AH
ESC      EQU     1BH
;
;
;STORAGE DEFINITION STATEMENTS
;
         ORG     DYNAT+0B80H   ;SYSTEM WORKING STORAGE AREA
PC1:     DS      2       ;USER'S PROGRAM COUNTER STORAGE
STKPTR:  DS      2       ;USER'S STACK POINTER STORAGE
BC:      DS      2       ;USER'S BC REG PR STORAGE
DE:      DS      2       ;USER'S DE REG PR STORAGE
```

```
HL:      DS      2       ;USER'S HL REG PR STORAGE
STSWRD:  DS      1       ;USER'S STATUS FLAG STORAGE
AC:      DS      1       ;USER'S ACCUMULATOR STORAGE
PC:      DS      2       ;USER'S PRIMARY PROGRAM COUNTER STORAGE
BINFLG:  DB      0       ;BINARY DISPLAY SWITCH(0=>NO BINARY DISPLAY)
CPOSIT:  DS      2       ;CURSER POSITION FOR ROLLUP PORTION OF SCREEN
STKTMP:  DS      2       ;TEMPORARY STORAGE FOR SYSTEM STACK
BASE:    DS      2       ;BASE ADDRESS STORAGE FOR VARIOUS ROUTINES
LAST:    DS      2       ;LAST    '    '    '    '    '
FROM:    DB      'FROM-' ;MESSAGES
TO:      DB      'TO-'   ;
ASCII:   DB      '0123456789ABCDEF'      ;ASCII HEX DIGIT TABLE
MOVENM:  DB      'MOV  ,  '      ;MOVE MNUMONIC
         DB      'COPYRIGHT (C) 1976, RICHARD E. MEADOR'
;
;CODE BEGINS HERE
;
         ORG     DYNAT   ;START ADDRESS
START:   LXI     SP,STACK        ;DEFINE SYSTEM STACK
         CALL    CLRSCR  ;CLEAR VIDEO SCREEN
         CALL    SETSCR  ;SET UP DISPLAY OF USER REGISTERS
         MVI     A,0     ;CLEAR ACCUMULATOR
         MVI     B,14    ;SET CLEAR COUNT
         LXI     H,PC1   ;SET FIRST ADDRESS TO BE CLEARED
VDM010:  MOV     M,A     ;CLEAR
         INX     H       ;NEXT ADDRESS
         DCR     B       ;1 LESS TO DO
         JNZ     VDM010  ;DONE?
         LXI     H,PC1       ;YES, DEFINE USER'S STACK POINTER
         SHLD    STKPTR      ;
VDM015:  CALL    DSREGS  ;DISPLAY CONTENTS OF USER'S REGISTERS
         CALL    KEYBDI  ;GET COMMAND
         CPI     40H     ;CHECK FOR ALPHA
         JM      VDM020  ;IGNORE IF NOT
         CPI     5BH     ;
         JP      VDM020  ;
         LXI     H,KEYTAB        ;GET COMMAND LOCATOR TABLE BASE ADDRESS
         SBI     40H     ;SUBTRACT ASCII BIAS FROM RECEIVED COMMAND
         RLC             ;DOUBLE FOR WORD INDEXING
         ADD     L       ;ADD INDEX
         MOV     L,A     ;
         MVI     A,0     ;
         ADC     H       ;
         MOV     H,A     ;
         MOV     E,M     ;GET COMMAND ADDRESS FROM TABLE
         INX     H       ;
         MOV     D,M     ;
         LXI     H,VDM020        ;SET UP RETURN ADDRESS
         PUSH    H       ;SAVE ON STACK
         XCHG            ;HL=COMMAND ROUTINE ADDRESS
```

DISK DRIVE WOES? PRINTER INTERACTION?
MEMORY LOSS? ERRATIC OPERATION?

DON'T BLAME THE SOFTWARE

Power Line Spikes, Surges & Hash could be the culprit!
Floppies, printers, memory & processor often interact!
Our unique ISOLATORS eliminate equipment interaction
AND curb damaging Power Line Spikes, Surges and Hash.

ESP Clear up Software and System problems with an ISOLATOR! ESP

ALL ISOLATORS: • 125 VAC, Standard 3-prong plug
- 1875 W MAX Load - 1 KW/Socket or socket bank
- Balanced Pi Filtered sockets or socket banks
- Spike/Surge Suppression - 1000 Amps, 8/20 usec

(SUPER ISOLATORS offer expended filtering and Spike/Surge Suppression capabilities)

ISO-1

ISO-2

ISO-1A	-3 individually filtered sockets	$ 56.95
ISO-4	-6 individually filtered sockets	96.95
ISO-2	-2 filtered banks; 6 sockets	56.95
ISO-5	-3 flitered banks; 9 sockets	79.95

IOS-6

ISO-7

*SWITCHABLE ISOLATORS - ALL ISOLATOR advantages combined with the versatility, convenience and utility of individually switched sockets. Each switch has associated pilot lite.

ISO-6	-3 switched, filtered sockets	$128.95
ISO-8	-5 switched, filtered sockets	161.95

*SUPER ISOLATORS - Cure for severe interference problems. Useful for Industrial applications and heavy duty controlled equipment or peripherals.

- Dual Balanced Pi Filtered sockets
- Spike/Surge Suppression - 2000 Amps, 8/20 usec

ISO-3	-3 super filtered sockets	$ 85.95
ISO-7	-5 Super-filtered sockets	139.95

*CIRCUIT BREAKER any model (add-CB) . ADD	7.00
*CKT BKR/SWITCH/PILOT any model (CBS) ADD	14.00

PHONE ORDERS 1-617-655-1532

ESP Electronic Specialists, Inc.
171 South Main Street, Natick, Mass. 01760

Dept.
696

6809 S-100 ads™

SINGLE BOARD COMPUTER

- MEETS I.E.E.E. S-100 STANDARD

MC 6809
- 10 addressing modes
- 24 indexed sub modes
- auto increment/decrement
- constant indexing from PC

- 4K/8K/16K ROM • 2K RAM
  ROM/RAM relocatable on 4K boundary
- ACIA; PIA; 8080 SIMULATED I/O
- 20 PARALLEL I/O LINES • 256 I/O PORTS
  ACIA provides RS-232 lines for asychronous communications with limited modem control at 8 selectable baud rates; I/O locatable at any 4K boundary
- COMPREHENSIVE MANUAL WITH SOFTWARE LISTINGS
- P.C. BOARD: SOLDERMASKED WITH PARTS LEGEND
  P.C. Board & Manual $69.95* + shipping
- adsMON: ADS MONITOR SUPPORTS BREAKPOINTS
  User definable interrupt service & more.

Available in PROM, write for prices.
Illinois residents add sales tax. *add $1.00 for shipping & handling

Ackerman Digital Systems, Inc.
110 N. York Rd., Suite 208, Elmhurst, Ill. 60126 (312) 530-8992

CM

computer mart
of new jersey

the microcomputer people®

THE VITAL INGREDIENT: EXPERTISE

Before you buy your new microcomputer, chances are you have a lot of questions. Important questions that could mean the difference between a working system and a wasted system. The vital ingredient is expertise. The microcomputer people at Computer Mart are expert at answering your questions and helping you put together the best system for your application. Whether it's for business, the home, or the laboratory, come see the experts at Computer Mart of New Jersey. We have the vital ingredient.

Computer Mart of New Jersey
501 Route 27
Iselin, N.J. 08830
(201) 283-0600

HOURS:
Open at 10 am,
Tuesday through Saturday

Dynatrace, cont'd...

```
        PCHL                    ;(THIS IS REALLY A CALL TO A COMMAND ROUTINE)
VDM020: LXI     H,LINE15        ;RESET CURSOR
        SHLD    CPOSIT  ;
        JMP     VDM015  ;END OF COMMAND PROCESSING LOOP
;
;COMMAND LOCATOR TABLE
;
KEYTAB: DW      UTURN   ;A-NOP
        DW      BINARY  ;B-BINARY DISPLAY TOGGLE
        DW      CALSUB  ;C-CALL USER SUB-PROGRAM
        DW      DUMPME  ;D-DUMP MEMORY TO SCREEN
        DW      UTURN   ;E-NOP
        DW      UTURN   ;F-NOP
        DW      GO      ;G-EXECUTE USER PROGRAM INTERPRETIVELY
        DW      UTURN   ;H-NOP
        DW      ISPEED  ;I-SET EXECUTION SPEED OF INTERPRETER
        DW      UTURN   ;J-NOP
        DW      UTURN   ;K-NOP
        DW      LOADRG  ;L-LOAD REGISTER PAIR
        DW      MEMSTR  ;M-STORE BYTES IN CONSECUTIVE MEMORY LOCATIONS
        DW      UTURN   ;N-NOP
        DW      UTURN   ;O-NOP
        DW      UTURN   ;P-NOP
        DW      UTURN   ;Q-NOP
        DW      READTP  ;R-READ PAPER TAPE
        DW      STEP    ;S-INTERPRET ONE INSTRUCTION
        DW      UTURN   ;T-NOP
        DW      UTURN   ;U-NOP
        DW      UTURN   ;V-NOP
        DW      UTURN   ;W-NOP
        DW      UTURN   ;X-NOP
        DW      UTURN   ;Y-NOP
        DW      ZERMEM  ;Z-ZERO MEMORY
;
;
;
UTURN:  RET             ;DUMMY ROUTINE FOR NOP'S
;
;
;  CLEAR VDM ROUTINE
;
CLRSCR: MVI     A,0     ;CONTROL PORT IINITIALIZATION WORD
        OUT     CONPRT  ;SEND
        LXI     B,SCREEN        ;BASE ADDRESS OF VDM BUFFER
CLR010: MVI     A,' '   ;ASCII SPACE
        STAX    B       ;CLEAR 1 BUFFER WORD
        INX     B       ;UPDATE BUFFER POINTER
        MVI     A,TOS AND 0FFH   ;LAST BUFFER ADDRESS +1
        CMP     B       ;CHECK FOR END OF BUFFER AREA
        JNZ     CLR010  ;DO UNTIL DONE
        LXI     H,LINE15        ;INITIAL CURSOR LOCATION
        SHLD    CPOSIT  ;STORE TO CURSOR SAVE
        RET             ;EXIT
;
;  SET DISPLAY OF USER'S REGISTERS
;
SETSCR: LXI     H,'PC'          ;PC
        SHLD    LINE1+5
        MVI     A,'A'           ;A
        STA     LINE1+20
        LXI     H,'BC'          ;BC
        SHLD    LINE1+34
        LXI     H,'IN'          ;IN
        SHLD    LINE2+2
        LXI     H,'ST'          ;ST
        SHLD    LINE2+4
        MVI     A,'R'           ;R
        STA     LINE2+6
        LXI     H,'DE'          ;DE
        SHLD    LINE2+34
        MVI     A,'C'           ;C
        STA     LINE3+15
        MVI     A,'Z'           ;Z
        STA     LINE3+9
        MVI     A,'P'           ;P
        STA     LINE3+13
        MVI     A,'S'           ;S
        STA     LINE3+8
        MVI     A,'A'           ;A
        STA     LINE3+11
        LXI     H,'HL'          ;HL
        SHLD    LINE3+34
        LXI     H,'SP'          ;SP
        SHLD    LINE4+34
        LXI     H,'PS'
        SHLD    LINE4+4
        MVI     A,'W'
        STA     LINE4+6
        MVI     A,'0'
        STA     LINE4+10
        STA     LINE4+12
        INR     A
        STA     LINE4+14
        RET
;
;MOVE THE NUMBER OF BYTES INDICATED IN DE
;FROM THE ADDRESS BEGINNING IN BC
;TO THE ADDRESS BEGINNING IN HL
;
SETLNE: LDAX    B       ;GETBYTE
        MOV     M,A     ;TRANSFER
        INX     H       ;UPDATE DESTINATION POINTER
        INX     B       ;UPDATE SOURCE POINTER
        DCX     D       ;UPDATE COUNT
        MOV     A,E     ;CHECK FOR COMPLETION
        CPI     0       ;
        JNZ     SETLNE  ;
        .•OV    A,D     ;
        CPI     0       ;
        JNZ     SETLNE  ;
        RET             ;EXIT
;
; DISPLAY REGISTERS ROUTINE
;
;THIS ROUTINE DECODES THE USER'S REGISTER INFORMATION
;     AND FORMATS IT FOR DISPLAY ON THE CRT SCREEN
;
DSREGS: PUSH    B       ;SAVE ALL REGISTERS
        PUSH    D       ;
        PUSH    H       ;
        PUSH    PSW     ;
```

```
        LXI     D,PC1   ;
        LXI     H,PCH+2 ;
        CALL    DSPHEX  ;
        LXI     D,PC1+1 ;
        LXI     H,PCH   ;
        CALL    DSPHEX  ;
        LXI     D,STKPTR    ;
        LXI     H,SPH+2 ;
        CALL    DSPHEX  ;
        LXI     D,STKPTR+1  ;
        LXI     H,SPH   ;
        CALL    DSPHEX  ;
        LXI     D,BC    ;
        LXI     H,BCH+2 ;
        CALL    DSPHEX  ;
        LXI     D,BC+1  ;
        LXI     H,BCH   ;
        CALL    DSPHEX  ;
        LXI     D,DE    ;
        LXI     H,DEH+2 ;
        CALL    DSPHEX  ;
        LXI     D,DE+1  ;
        LXI     H,DEH   ;
        CALL    DSPHEX  ;
        LXI     D,HL    ;
        LXI     H,HLH+2 ;
        CALL    DSPHEX  ;
        LXI     D,HL+1  ;
        LXI     H,HLH   ;
        CALL    DSPHEX  ;
        LXI     D,AC    ;
        LXI     H,ACCH  ;
        CALL    DSPHEX  ;
        LDA     STSWRD  ;
        LXI     H,CYB   ;
        RAR             ;
        MVI     M,'0'   ;
        JNC     DSR010  ;
        MVI     M,'1'   ;
DSR010: RAR             ;
        RAR             ;
        LXI     H,PB    ;
        MVI     M,'0'   ;
        JNC     DSR020  ;
        MVI     M,'1'   ;
DSR020: RAR             ;
        RAR             ;
        LXI     H,ACB   ;
        MVI     M,'0'   ;
        JNC     DSR030  ;
        MVI     M,'1'   ;
DSR030: RAR             ;
        RAR             ;
        LXI     H,ZB    ;
        MVI     M,'0'   ;
        JNC     DSR040  ;
        MVI     M,'1'   ;
DSR040: RAR             ;
        LXI     H,SB    ;
        MVI     M,'0'   ;
        JNC     DSR050  ;
        MVI     M,'1'   ;
DSR050: POP     PSW     ;
        POP     H       ;
        POP     D       ;
        POP     B       ;
        LDA     BINFLG  ;CHECK FOR BINARY DISPLAY FLAG SET
        ANA     A       ;
        RZ              ;DONE IF NOT SET
        PUSH    B       ;
        PUSH    D       ;
        PUSH    H       ;
        PUSH    PSW     ;
        LXI     D,AC    ;
        LXI     H,ACCB  ;
        CALL    DSPBIN  ;
        LXI     D,BC    ;
        LXI     H,BCB+9 ;
        CALL    DSPBIN  ;
        LXI     D,BC+1  ;
        LXI     H,BCB   ;
        CALL    DSPBIN  ;
        LXI     D,DE    ;
        LXI     H,DEB+9 ;
        CALL    DSPBIN  ;
        LXI     D,DE+1  ;
        LXI     H,DEB   ;
        CALL    DSPBIN  ;
        LXI     D,HL    ;
        LXI     H,HLB+9 ;
        CALL    DSPBIN  ;
        LXI     D,HL+1  ;
        LXI     H,HLB   ;
        CALL    DSPBIN  ;
        POP     PSW     ;
        POP     H       ;
        POP     D       ;
        POP     B       ;
        RET             ;
;
; DECODE REGISTERS FOR DISPLAY ROUTINE
;
DSPHEX: LDAX    D       ;GET REGISTER CONTENTS
        RRC             ;SCALE ACCUMULATOR DOWN TO GET UPPER HEX DIGIT
        RRC             ;
        RRC             ;
        RRC             ;
        ANI     0FH     ;MASK OFF UNWANTED BITS
        LXI     B,ASCII ;BASE ADDRESS OF ASCIIDIGIT TABLE
        ADD     C       ;ADD ACCUMULATOR TO BC TO GET ADDRESS OF DIGIT
        MOV     C,A     ;
        MVI     A,0     ;
        ADC     B       ;
        MOV     B,A     ;
        LDAX    B       ;
        MOV     M,A     ;STORE ASCII CODE FOR HEX DIGIT
        INX     H       ;INCREMENT SCREEN POSITION
        LDAX    D       ;GET REGISTER CONTENTS AGAIN
        ANI     0FH     ;DO THE SAME FOR THE LOWER HEX DIGIT
        LXI     B,ASCII ;
        ADD     C       ;
        MOV     C,A     ;
        MVI     A,0     ;
        ADC     B       ;
        MOV     B,A     ;
        LDAX    B       ;
        MOV     M,A     ;
        RET             ;
;
; DECODE REGISTER TO BINARY DISPLAY ROUTINE
;
DSPBIN: LDAX    D       ;GET USER REGISTER CONTENTS
        MVI     C,8     ;# OF BITS
DSP010: RAL             ;SHIFT UPPER BIT TO CARRY FOR CHECKING
        MVI     M,'0'   ;ASSUME ZERO
        JNC     DSP020  ;CHECK FOR ONE
        MVI     M,'1'   ;CHANGE IF ONE
DSP020: INX     H       ;MOVE TO NEXT SCREEN POSITION
        DCR     C       ;COUNT OFF
        JNZ     DSP010  ;DONE?
        RET             ;YES
;
;SIMULATOR CODE BEGINS HERE
;  ONE 8080 INSTRUCTION AS INDICATED BY THE USER'S PC REGISTER
STEP:   LHLD    PC      ;GET USER'S PROGRAM COUNTER VALUE
        SHLD    PC1     ;SAVE FOR LAGGING DISPLAY
        MOV     A,M     ;GET THE CONTENTS OF MEMORY LOCATION INDICATED
        CPI     40H     ;CHECK FOR CODES 0-3F HEX
        JC      STP010  ;LOW ORDER 64 OPCODES?
        CPI     80H     ;NO, CHECK FOR 40-7F HEX
        JNC     STP020  ;IS A REGISTER TO REGISTER MOVE INSTRUCTION?
        CALL    MOVE    ;YES, EXECUTE IT
        RET             ;RETURN TO MONITOR
STP010: CALL    OPLOW   ;LOW ORDER OP CODES
        RET             ;RETURN TO MONITOR
STP020: CPI     0C0H    ;CHECK FOR 80-BF HEX
        JNC     STP030  ;ARTHMETIC INSTRUCTION?
        CALL    ARITH   ;YES, EXECUTE
        RET             ;RETURN TO MONITOR
STP030: CALL    OPHIGH  ;HIGH ORDER OP CODE
        RET             ;RETURN TO MONITOR
;
; EXECUTE INSTRUCTION ROUTINE
;
OPEXEC: LXI     H,0     ;SAVE MONITOR'S STACK POINTER
        DAD     SP
        SHLD    STKTMP
        LHLD    STSWRD . ;LOAD REAL REGISTERS WITH USER'S DATA
        PUSH    H
        POP     PSW
        LHLD    STKPTR
        SPHL
        LHLD    BC
        MOV     B,H
        MOV     C,L
        LHLD    DE
        XCHG
        LHLD    HL
INSTR:  DS      3       ;INSTRUCTION TO BE EXECUTED IS STORED HERE
                        ; IF IT WILL NOT CAUSE LOSS OF CONTROL
        SHLD    HL      ;SAVE USER REGISTER VALUES
        PUSH    PSW     ;
        POP     H       ;
        SHLD    STSWRD  ;
        LXI     H,0     ;
        DAD     SP      ;
        SHLD    STKPTR      ;
        XCHG            ;
        SHLD    DE      ;
        LHLD    STKTMP  ;GET MONITOR'S STACK POINTER
        SPHL            ;
        MOV     H,B     ;
        MOV     L,C     ;
        SHLD    BC      ;
        RET             ;
;
; MOV COMMAND EXECUTION ROUTINE
;
MOVE:   MOV     B,A     ;SAVE OP CODE
        ANI     07H     ;DECODE SOURCE REGISTER
        MVI     D,0     ;
        MOV     E,A     ;
        LXI     H,RGSTRS        ;
        DAD     D       ;
        MOV     A,M     ;GET REGISTER NAME
        STA     MOVENM+6   ;SET UP NMEUMONIC
        MOV     A,B     ;RESTORE OP CODE
        RRC             ;
        RRC             ;
        RRC             ;
        ANI     07H     ;DO SAME FOR DESTINATION REGISTER
        MOV     E,A     ;
        LXI     H,RGSTRS        ;
        DAD     D       ;
        MOV     A,M     ;
        STA     MOVENM+4   ;
        MOV     A,B     ;
        MVI     B,1     ;# OF BYTES IN INSTRUCTION
        CALL    MVINST  ;COPY INSTRUCTION INTO EXECUTION AREA
        LXI     H,INSTA ;DISPLAY NMEUMONIC
        LXI     B,MOVENM   ;
        LXI     D,8     ;
        CALL    SETLNE  ;
        CALL    OPEXEC  ;GO EXECUTE INSTRUCTION
        RET             ;DONE,
;
; ARITHMETIC INSTRUCTION EXECUTION ROUTINE
;
ARITH:  MOV     B,A     ;SAVE OP CODE
        LXI     H,ARITHN        ;GET ADDRESS OF ARITH NMEUMONICS
        ANI     38H     ;ISOLATE REGISTER OPERAND
        MVI     D,0     ;
        MOV     E,A     ;
        DAD     D       ;ADD INDEX TO GET APPROPRIATE NMEUMONIC
        PUSH    H       ;SAVE TEMPORARILY
        MOV     A,B     ;RESTORE OPCODE
        ANI     07H     ;ISOLATE REGISTER OPERAND # .
        LXI     H,RGSTRS        ;GET REGISTER NAME LIST ADDRESS
        MOV     E,A     ;
        DAD     D       ;ADD INDEX TO GET CORRECT REGISTER NMEUMONIC
        MOV     A,M     ;GET NMEUMONIC
        POP     H       ;RETRIEVE INSTRUCTION NMEUMONIC ADDRESS
        MVI     E,4     ;
        DAD     D       ;
        MOV     M,A     ;MOVE REGISTER NAME TO MEMORY
```

```
        DCX     SP      ;
        DCX     SP      ;
        MOV     A,B     ;RESTORE OP CODE
        MVI     B,1     ;SET # OF BYTES IN OPCODE
        CALL    MVINST  ;MOVE INSTRUCTION TO EXECUTION AREA
        LXI     H,INSTA ;DISPLAY NMEUMONIC
        LXI     D,8     ;
        POP     B       ;
        CALL    SETLNE  ;
        CALL    OPEXEC  ;GO EXECUTE INSTRUCTION
        RET             ;
;
;ARITHMETIC INSTRUCTION NMEUMONICS
;
ARITHN: DB      'ADD    '       ;ADD REGISTER TO ACCUMULATOR
        DB      'ADC    '       ;ADD REGISTER+CARRY TO ACCUMULATOR
        DB      'SUB    '       ;SUB REGISTER FROM ACCUMULATOR
        DB      'SBB    '       ;SUB REGISTER+CARRY FROM ACCUMULATOR
        DB      'ANA    '       ;LOGICAL AND REGISTER WITH ACCUMULATOR
        DB      'XRA    '       ;LOGICAL EXCLUSIVE OR REGISTER WITH ACCU
        DB      'ORA    '       ;LOGICAL OR REGISTER WITH ACCUMULATOR
        DB      'CMP    '       ;COMPARE REGISTER WITH ACCUMULATOR
;
; LOW ORDER INSTRUCTION EXECUTION ROUINE
; OP CODES 0-3F HEX
OPLOW:  MOV     E,A     ;SAVE OP CODE
        ANI     07H     ;DETERMINE CLASS AND BRANCH TO APPRPRIATE ROUTINE
        RLC             ;
        MOV     C,A     ;
        MVI     B,0     ;
        LXI     H,OPTAB ;
        DAD     B       ;
        MOV     A,E     ;
        MOV     E,M     ;
        INX     H       ;
        MOV     D,M     ;
        XCHG            ;
        PCHL            ;
;
;OP CODE SIMULATION ROUTINE LOOKUP TABLE
;
OPTAB:  DW      NPINST    ; NO-OP ROUTINE POINTER
        DW      LXIDAD  ; LXI & DAD ROUTINE POINTER
        DW      LDSTRX  ; LDAX & STAX ROUTIN POINTER
        DW      INXDCX  ;INX & DCX ROUTINE POINTER
        DW      INRDCR  ; INR & DCR ROUTINE POINTER
        DW      INRDCR  ; SAME
        DW      MVIMM     ; MVI ROUTINE POINTER
        DW      ROTATE  ; ROTATE POINTER
;
; A=OP CODE, B= NO. OF BYTES
;
MVINST: STA     INSTR   ;STORE OP CODE TO EXECUTION AREA
        PUSH    B       ;SAVE NUMBER OF BYTES IN OP CODE
        CALL    HEXASC  ;CONVERT OP CODE TO ASCII/HEX EQUIVALANT
        MOV     H,C     ;TRANSFER FOR DOUBLE LENGTH STORE
        MOV     L,B     ;
        SHLD    INSTH   ;STORE ASCII CODES OF OP CODE TO VDM BUFFER
        POP     B       ;RETRIEVE NUMBER OF BYTES IN INSTRUCTION
        LXI     D,INSTR+1       ;GET ADDRESS OF NEXT LOCATION IN EXECUTI
        MVI     A,0     ;ZERO NEXT 2 BYTES IN EXECUTION AREA
        STAX    D       ;    IN CASE INSTRUCTION<3 BYTES
        STA     INSTR+2 ;
        MVI     H,' '   ;CLEAR PREVIOUS EXECUTION FROM SCREEN
        MOV     L,H     ;
        SHLD    INSTH+2 ;
        SHLD    INSTH+4 ;
        LXI     H,INSTH+2       ;SET TO DISPLAY REST OF INSTRUCTION IF A
        SHLD    STKTMP  ;SAVE VDM BUFFER LOCATION OF ASCII/HEX DISPLAY A
        LHLD    PC      ;GET USER'S PROGRAM COUNTER
        INX     H       ;UPDATE IT
MVI010: DCR     B       ;COUNT OFF NUMBER OF BYTES IN INSTRUCTION
        JZ      MVI020  ;EXIT IF DONE
        MOV     A,M     ;GET NEXT BYTE OF INSTRUCTION FROM USER'S PROGRA
        INX     H       ;UPDATE USER'S PROGRAM COUNTER
        STAX    D       ;STORE BYTE TO EXECUTION AREA
        INX     D       ;UPDATE EXECUTION BUFFER POINTER
        PUSH    B       ;SAVE NUMBER OF BYTES IN INSTRUCTION
        PUSH    D       ;SAVE EXECUTION BUFFER POINTER
        PUSH    H       ;SAVE USER'S PROGRAM COUNTER
        CALL    HEXASC  ;CONVERT NEXT BYTE OF INSTRUCTION TO ASCII/HEX C
        LHLD    STKTMP  ;RETRIEVE VDM BUFFER LOCATIN OF ASCII/HEX DISPLA
        MOV     M,B     ;MOVE ASCII/HEX CODES OF CURRENT INSTRUCTION BYT
        INX     H       ;   TO VDM BUFFER AREA
        MOV     M,C     ;
        INX     H       ;
        SHLD    STKTMP  ;SAVE VDM BUFFER LOCATION AGAIN
        POP     H       ;RETRIEVE USER'S PROGRAM COUNTER
        POP     D       ;RETRIEVE EXECUTION BUFFER POINTER
        POP     B       ;RETRIEVE NUMBER OF BYTES LEFT IN INSTRUCTION
        JMP     MVI010  ;CONTINUE UNTIL ALL BYTES MOVED
MVI020: SHLD    PC      ;STORE UPDATED USER'S PROGRAM COUNTER
        RET             ;EXIT
;
;
; MOVE NMUMONIC TO BUFFER
;
MVNMNC: LXI     H,INSTA ;GET VDM BUFFER LOCATION OF MNEUMONIC DISPLAY AR
        PUSH    D       ;SAVE D REGISTER
        LXI     D,4     ;SET NUMBER OF BYTES TO TRANSFER
        CALL    SETLNE  ;DO TRANSFER
        POP     B       ;GET OLD D REGISTER
        MVI     D,' '   ;THE FOLLOWING CODE MOVES THE ASCII CHARACTERS O
        MOV     M,D     ;   THE REGISTERS USED BY THE INSTRUCTION TO THE
        INX     H       ;       THE VDM BUFFER AREA
        MOV     M,C     ;
        INX     H       ;
        MOV     M,B     ;
        INX     H       ;
        MOV     M,D     ;
        RET             ;
;
; GET REGISTER PAIR ROUTINE
;
GETRP:  ANI     30H     ;MASK REGISTER PAIR FIELD
        RRC             ;SCALE FOR INDEXING
        RRC             ;
        RRC             ;
        MOV     E,A     ;BUILD INDEX IN DE
        MVI     D,0     ;
        LXI     H,REGPAR        ;GET BASE ADDRESS OF REG PAIR MNEUMONICS
        DAD     D       ;ADD INDEX
        SHLD    GET010+1        ;SAVE POINTER
GET010: LHLD    0       ;0 IS REPLACED BY POINTER FROM ABOVE
```

Dynatrace, cont'd...

```
        XCHG             ;POSITION FOR SUB ROUTINE CALL
        CALL    MVNMNC   ;DISPLAY OP CODE
        RET              ;
;
; EXECUTE NO-OP ROUTINE
;
NPINST:    MVI     B,1      ;SET INSTRUCTION LENGTH
        CALL    MVINST   ;MOVE INSTRUCTION TO EXECUTION AREA
        LXI     D,'  '   ;NO REGISTER DATA
        LXI     B,LOWOP  ;ADDRESS OF NOP MNEUMONIC
        CALL    MVNMNC   ;DISPLAY OP CODE
        CALL    OPEXEC   ;EXECUTE INSTRUCTION
        RET              ;
;
;  EXECUTE LXI AND DAD INSTRUCTIONS ROUTINE
;
LXIDAD: MOV     B,A      ;SAVE INSTRUCTION
        ANI     08H      ;ISOLATE LXI/DAD BIT
        MOV     A,B      ;RESTORE INSTRUCTION
        MVI     B,1      ;ASSUME DAD INSTRUCTION
        JNZ     LXI010   ;DAD OR LXI?
        MVI     B,3      ;LXI, CHANGE INSTRUCTION LENGTH TO 3 BYTES
LXI010: PUSH    PSW      ;SAVE INSTRUCTION DESIGNATOR WHICH IS THE ZERO F
        CALL    MVINST   ;DISPLAY OP CODE
        POP     PSW      ;RETRIEVE FLAG
        LXI     B,LOWOP1+4        ;ASSUME DAD
        JNZ     LXI020   ;DAD/LXI?
        LXI     B,LOWOP1          ;LXI, CHANGE POINTER
LXI020: CALL    GETRP    ;ISOLATE REG PAIR
        CALL    OPEXEC   ;EXECUTE INSTRUCTION
        RET              ;
;
; LDA STA LDAX STAX SHLD LHLD  EXECUTION ROUTINE
;
LDSTRX: MOV     C,A      ;SAVE INSTRUCTION
        ANI     20H      ;ISOLATE (LDX,STX)/(LHLD,SHLD,LDA,STA) BIT
        MOV     A,C      ;RESTORE INSTRUCTION
        JNZ     LDS020   ;(LDX,STX)/(LHLD,SHLD,LDA,STA)?
        ANI     08H      ;ISOLATE LDX/STX BIT
        MOV     A,C      ;RESTORE INSTRUCTION
        MVI     B,1      ;INSTRUCTION LENGTH IS 1 FOR BOTH
        PUSH    PSW      ;SAVE DESIGNATOR BIT
        CALL    MVINST   ;MOVE INSTRUCTION TO EXECUTION AREA
        POP     PSW      ;RETRIEVE DESIGNATOR BIT
        LXI     B,LOWOP2          ;ASSUME STAX
        JZ      LDS010   ;STAX/LDAX?
        LXI     B,LOWOP2+4        ;LDAX, CHANGE MNEUMONIC POINTER
LDS010: CALL    GETRP    ;DETERMINE REGISTER PAIR
        JMP     LDS030   ;GO EXECUTE
LDS020: MVI     B,3      ;(LHLD,SHLD,LDA,STA), SET LENGTH TO 3 BYTES
        CALL    MVINST   ;MOVE INSTRUCTION TO EXECUTION AREA
        MOV     A,C      ;RESTORE INSTRUCTION
        ANI     18H      ;ISOLATE INSTRUCTION DESIGNATOR
        RRC              ;SCALE FOR INDEXING
        MOV     E,A      ;BUILD INDEX
        MVI     D,0      ;
        LXI     H,LOWOP2+8        ;GET BASE ADDRESS OF INSTRUCTION GROUP
        DAD     D        ;ADD INDEX
        MOV     B,H      ;POSITION FOR CALL
        MOV     C,L      ;
        LXI     D,'  '   ;SET REG TYPE TO NULL
        CALL    MVNMNC   ;DISPLAY CODE
LDS030: CALL    OPEXEC   ;EXECUTE INSTRUCTION
        RET              ;
;
; INX DCX EXECUTION ROUTINE
;
INXDCX: MOV     C,A      ;SAVE INSTRUCTION
        MVI     B,1      ;THESE ARE ALL ONE BYTE
        CALL    MVINST   ;MOVE INSTRUCTION TO EXECUTION AREA
        MOV     A,C      ;RESTORE INSTRUCTION
        PUSH    PSW      ;SAVE IT AGAIN
        ANI     08H      ;ISOLATE DESIGNATOR BIT
        LXI     B,LOWOP3          ;ASSUME INX
        JZ      INX010   ;INX/DCX?
        LXI     B,LOWOP3+4        ;DCX, CHAINGE POINTER
INX010: POP     PSW      ;GET INSTRUCTION BACK
        CALL    GETRP    ;DETERMINE REG PAIR
        CALL    OPEXEC   ;EXECUTE INSTRUCTION
        RET              ;
;
; INR DCR EXECUTION ROUTINE
;
INRDCR: MOV     C,A      ;SAVE INSTRUCTION
        MVI     B,1      ;THESE ARE ALL ONE BYTE
        CALL    MVINST   ;MOVE INSTRUCTION TO EXECUTION AREA
        MOV     A,C      ;RESTORE INSTRUCTION
        STA     STKTMP   ;SAVE IT AGAIN
        LXI     B,LOWOP4          ;ASSUME INR
        ANI     01H      ;CHECK DESIGNATOR BIT
        JZ      INR010   ;INR/DCR?
        LXI     B,LOWOP5          ;DCR, CHANGE POINTER
INR010: LDA     STKTMP   ;GET INSTRUCTION BACK
        CALL    GETREG   ;DETERMINE REGISTER
        CALL    OPEXEC   ;EXECUTE INSTRUCTION
        RET              ;
;
; MVI EXECUTION ROUTINE
;
MVIMM:     MOV     C,A      ;SAVE INSTRUCTION
        MVI     B,2      ;THESE ARE ALL 2 BYTE INSTRUCTIONS
        CALL    MVINST   ;MOVE INSTRUCTION TO EXECUTION AREA
        MOV     A,C      ;RESTORE INSTRUCTION
        LXI     B,LOWOP6          ;SET POINTER MVI MNEUMONIC
        CALL    GETREG   ;DETERMINE REGISTER
        CALL    OPEXEC   ;EXECUTE INSTRUCTION
        RET              ;
;
; RLC RRC RAL RAR DAA CMA STC CMC
; EXECUTION ROUTINE
;
ROTATE: MOV     C,A      ;SAVE INSTRUCTION
        MVI     B,1      ;THESE ARE ALL ONE BYTE INSTRUCTIONS
        CALL    MVINST   ;MOVE INSTRUCTION TO EXECUTION AREA
        MOV     A,C      ;RESTORE INSTRUCTION
        ANI     38H      ;ISOLATE DESIGNATOR BITS
        RRC              ;SCALE FOR INDEXING
        MOV     E,A      ;CONSTRUCT INDEX
        MVI     D,0      ;
        LXI     H,LOWOP7          ;GET BASE OF ADDRESS OF MNEUMONICS
        DAD     D        ;ADD INDEX
```

```
        MOV     C,L      ;POSITION ADDRESS FOR SUBROUTINE CALL
        MOV     B,H      ;
        LXI     D,'  '   ;SET REGISTER TO NULL
        CALL    MVNMNC   ;DISPLAY OP CODE
        CALL    OPEXEC   ;EXECUTE INSTRUCTION
        RET              ;
;
; GET REGISTER ROUTINE
;
GETREG: ANI     38H      ;ISOLATE REGISTER DESIGNATOR BITS
        RRC              ;SCALE FOR INDEXING
        RRC              ;
        RRC              ;
        MOV     E,A      ;CONSTRUCT INDEX WORD
        MVI     D,0      ;
        LXI     H,RGSTRS          ;GET BASE ADDRESS OF REGISTER CODES
        DAD     D        ;ADD INDEX
        MVI     E,' '    ;OTHER REGISTER IS NULL
        MOV     D,M      ;GET THIS REGISTER
        CALL    MVNMNC   ;DISPLAY OP CODE AND REGISTER
        RET              ;
LOWOP:  DB      'NOP '   ;NO-OP
LOWOP1: DB      'LXI '   ;LOAD REGISTER PAIR IMMEDIATE(EXTENDED)
        DB      'DAD '   ;DOUBLE LENGTH ADD
LOWOP2: DB      'STAX'   ;STORE ACCUMULATOR INDIRECT THROUGH REG PAIR
        DB      'LDAX'   ;LOAD ACCUMULATOR INDIRECT THROUGH REG PAIR
        DB      'SHLD'   ;STORE HL DIRECT
        DB      'LHLD'   ;LOAD HL DIRECT
        DB      'STA '   ;STORE ACCUMULATOR DIRECT
        DB      'LDA '   ;LOAD ACCUMULATOR DIRECT
LOWOP3: DB      'INX '   ;INCREMENT REGISTER PAIR
        DB      'DCX '   ;DECREMENT REGISTER PAIR
LOWOP4: DB      'INR '   ;INCREMENT REGISTER
LOWOP5: DB      'DCR '   ;DECREMENT REGISTER
LOWOP6: DB      'MVI '   ;MOVE DATA IMMEDIATE TO REGISTER
LOWOP7: DB      'RLC '   ;ROTATE LEFT THROUGH CARRY
        DB      'RRC '   ;ROTATE RIGHT THROUGH CARRY
        DB      'RAL '   ;ROTATE ARITHMETIC LEFT
        DB      'RAR '   ;ROTATE ARITHMETIC RIGHT
        DB      'DAA '   ;DECIMAL ADJUST ACCUMULATOR
        DB      'CMA '   ;COMPLIMENT ACCUMULATOR
        DB      'STC '   ;SET CARRY
        DB      'CMC '   ;COMPLIMENT CARRY
RGSTRS: DB      'BCDEHLMA'        ;REGISTER CODES
REGPAR: DB      'B D H SP'        ;REGISTER PAIR CODES
FLAGS:  DB      'NZZ NCC POPEP M '        ;STATUS FLAG CODES
;
;
;
KEYBDI: PUSH    H        ;SAVE HL
        LHLD    CPOSIT   ;GET CURSOE POSITION
KEY005: IN      KSTAT  ;CHECK 3P+S STATUS
        ANI     KBDRDY   ;LOOK FOR KEY BOARD READY
        JNZ     KEY005   ;LOOP ON NO DATA AVAILABLE
        IN      KEYBRD   ;GET DATA FROM ASCII KEYBOARD
        ANI     7FH      ;STRIP OFF PARITY BIT
        MOV     B,A      ;SAVE CHARACTER
        CPI     ESC      ;CHECK FOR ESCAPE SEQUENCE
        JNZ     KEY007   ;WAS IT?
        LXI     SP,STACK          ;YES, RESET STACK POINTER
        CALL    BMPLNE   ;SCROLL UP ONE LINE
        JMP     VDM015   ;RESUME SCAN LOOP
KEY007: CPI     CR       ;CHECK FOR CARRIAGE RETURN
        JZ      KEY020   ;WAS IT
KEY010: MOV     M,A      ;NO, DISPLAY CHARACTER
        INX     H        ;UPDATE CURSER
        MOV     A,H      ;CHECK FOR VDM BUFFER OVERFLOW
        CPI     TOS AND OFFH     ;
        JC      KEY030   ;
KEY020: CALL    BMPLNE   ;SCROLL UP ON BUFFER FULL
        LXI     H,LINE15          ;RESET CURSOR TO BEGINNING OF LAST LINE
KEY030: MOV     A,B      ;RESTORE CHARATER RECEIVED FROM KEYBOARD
        SHLD    CPOSIT   ;STORE UPDATED CURSOR
        POP     H        ;RESTORE ORIGINAL CONTENTS OF HL
        RET              ;
;
;
;
BMPLNE: PUSH    H        ;SAVE HL
        PUSH    D        ;SAVE DE
        PUSH    B        ;SAVE BC
        PUSH    PSW      ;SAVE PSW
        LXI     H,MIDSCR          ;BUFFER ADDRESS TO MOVE OLD DATA TO
        LXI     B,MIDSCR+LINE     ;BUFFER ADDRESS TO GET OLD DATA FROM
        MVI     E,8      ;NUMBER OF LINES TO BE SCROLLED
BMP010: PUSH    D        ;SAVE
        LXI     D,64     ;NUMBER OF CHARACTERS PER LINE
        CALL    SETLNE   ;SCROLL ONE LINE AT A TIME FROM THE TOP DOWN
        POP     D        ;GET NUMBER OF LINES LEFT TO GO
        DCR     E        ;COUNT OFF ONE MORE
        JNZ     BMP010   ;DO UNTIL DONE
        MVI     A,' '    ;STE TO SPACE OUT LAST LINE
        MVI     D,64     ;BLANK OUT ALL CHARACTERS ON LAST LINE
        LXI     H,LINE15          ;GET ADDRESS OF FIRST CHARACTER OF LAST
BMP020: MOV     M,A      ;BLANK IT OUT
        INX     H        ;UPDATE POINTER
        DCR     D        ;COUNT DOWN
        JNZ     BMP020   ;DO WHILE NOT DONE
        LXI     H,LINE15          ;RESET CURSOR
        SHLD    CPOSIT   ;
        POP     PSW      ;RESTORE ORIGINAL REGISTER CONTENTS
        POP     B        ;
        POP     D        ;
        POP     H        ;
        RET              ;
;
;
;
BINARY: LDA     BINFLG   ;GET BINARY DISPLAY FLAG
        CMA              ;TOGGLE IT
        STA     BINFLG   ;PUT IT BACK
        CALL    CLRSCR   ;RESET SCREEN
        CALL    SETSCR   ;
        RET              ;
;
;
;
GTLMTS: CALL    BMPLNE   ;SCROLL UP
        LXI     B,FROM   ;DISPLAY FROM MESSAGE
        LXI     H,LINE15          ;
        LXI     D,5      ;
        CALL    SETLNE   ;
        LXI     H,LINE15+5        ;
        SHLD    CPOSIT   ;
        CALL    GETADR   ;GET ONE HEX NUMBER OF UP TO 4 DIGITS
```

```
SHLD    BASE    ;THAT WAS THE BASE ADDRESS
CALL    BMPLNE  ;SCROLL UP
        LXI     B,TO    ;DISPLAY TO MESSAGE
        LXI     H,LINE15        ;
        LXI     D,3     ;
        CALL    SETLNE  ;
        LXI     H,LINE15+3      ;
        SHLD    CPOSIT  ;
        CALL    GETADR  ;GET ONE HEX NUMBER
        XCHG            ;MOVE TO DE
        LHLD    BASE    ;GET BASE ADDRESS
        CALL    BMPLNE  ;SCROLL UP
        RET             ;
;
;
;
DUMPME: CALL    GTLMTS  ;GET DUMP LIMITS
DMP010: PUSH    D       ;SAVE LAST ADDRESS
        CALL    DMEM16  ;DISPLAY 16 BYTES
        POP     D       ;GET LAST ADDRESS
        MOV     A,E     ;SEE IF LAST ADDRESS EXCEEDED
        SUB     L       ;
        MOV     A,D     ;
        SBB     H       ;
        JP      DMP010  ;CONTINUE DUMPING IF NOT
        RET             ;
;
;
;
GETBYT: MVI     E,2     ;GET TWO CHARACTERS FROM KEY BOARD
        JMP     GTA005  ;GO AROUND NEXT ENTRY POINT
GETADR: MVI     E,4     ;GET 4 CHARACTERSFROM KEYBOARD
GTA005: LXI     H,0     ;START WITH ZERO
GTA010: CALL    KEYBDI  ;GET ACHARACTER FROM THE KEYBOARD
        CPI     CR      ;CHECK FOR TERMINATOR
        JZ      GTA020  ;EXIT ON END OF NUMBER
        CALL    ASCHEX  ;CONVERT CHARACTER TO HEX DIGIT
        MOV     C,A     ;POISITION FOR ADD
        MVI     B,0     ;
        DAD     H       ;MULTIPLY HL BY 16
        DAD     H       ;
        DAD     H       ;
        DAD     H       ;
        DAD     B       ;ADD LAST CHARACTER RECIEVED
        DCR     E       ;COUNT OFF NUMBER OF DIGITS
        JNZ     GTA010  ;DO UNTIL COUNT EXAUSTED OR CARRIAGE RETURN
GTA020: RET             ;
;
;
;
ASCHEX: SBI     '0'     ;SUBTRACT OFF ASCII BIAS
        JM      ASC010  ;CHAR<0?
        CPI     10      ;NO,
        RC              ;DONE IF CHAR<=9
        SBI     07H     ;ADJUST FOR ALPHA BIAS
        CPI     LF      ;
        JC      ASC010  ;
        CPI     10H     ;CHAR>0FH?
        RC              ;NO, RETURN
ASC010: MVI     A,0     ;RETURN ZERO IF ILLEAGLE
        RET             ;
;
;DUMP 16 BYTES TO TTY
;
DMEM16: MOV     A,L     ;DISPLAY ADDRESS
        CALL    HEXASC  ;
        PUSH    B       ;
        MOV     A,H     ;
        CALL    HEXASC  ;
        LXI     D,LINE15        ;
        XCHG            ;
        MOV     M,B     ;
        INX     H       ;
        MOV     M,C     ;
        INX     H       ;
        POP     B       ;
        MOV     M,B     ;
        INX     H       ;
        MOV     M,C     ;
        MVI     B,16    ;NUMBER OF BYTES TO DISPLAY
DME010: PUSH    B       ;SAVE
        INX     H       ;UPDATE ADDRESS
        LDAX    D       ;GET BYTE FROM MEMORY
        INX     D       ;UPDATE POINTER
        CALL    HEXASC  ;CONVERT TO ASCII CHARACTERS AND DISPLAY
        INX     H       ;
        MOV     M,B     ;
        INX     H       ;
        MOV     M,C     ;
        POP     B       ;
        DCR     B       ;
        JNZ     DME010  ;DO UNTIL ALL 16 DISPLAYED
        CALL    BMPLNE  ;SCROLL UP
        XCHG            ;
        RET             ;
;
;CONVERT BYTE IN ACCUMULATOR TO ASCII DIGITS
;
HEXASC: PUSH    D       ;SAVE DE
        PUSH    H       ;SAVE HL
        LXI     H,ASCII ;GET BASE ADDRESS OF ASCII CODE
        MOV     B,A     ;SAVE DATA
        ANI     0FH     ;DO LOWER HEX DIGIT FIRST
        MOV     E,A     ;SET UP FOR INDEXING
        MVI     D,0     ;
        DAD     D       ;ADD INDEX
        MOV     C,M     ;GET CHARACTER
        MOV     A,B     ;RESTORE DATA
        ANI     0F0H    ;DO UPPER HEX DIGIT
        RRC             ;SCALE DOWN FOR INDEXING
        RRC             ;
        RRC             ;
        RRC             ;
        MOV     E,A     ;SET UP FOR INDEXING
        LXI     H,ASCII ;GET BASE ADDRESS OF ASCII CODES AGAIN
        DAD     D       ;ADD INDEX
        MOV     B,M     ;GET CHARACTER
        POP     H       ;RESTORE HL
        POP     D       ;RESTORE DE
        RET             ;
;
```

```
;LOAD REGISTERS FROM KEYBOARD ROUTINE
;
LOADRG: CALL    KEYBDI  ;GET REGISTER IDENTIFIER
        LXI     H,LDREGS        ;DETERMINE SELECTED REGISTER
        MVI     D,6     ;6 POSSIBLE REGISTER PAIRS
        MOV     B,A     ;SAVE REGISTER DESIGNATOR
LOA010: MOV     A,M     ;GET ONE FROM MEMORY
        CMP     B       ;CHECK AGAINST SELECTED REGISTER PAIR
        JZ      LOA020  ;MATCH
        INX     H       ;NO, KEEP LOOKING
        DCR     D       ;
        JNZ     LOA010  ;
        RET             ;RETURN IF NOT FOUND
LOA020: MOV     A,D     ;GET INDEX NUMBER
        DCR     A       ;ADJUST
        RLC             ;SCALE
        MOV     E,A     ;SET UP INDEX
        MVI     D,0     ;
        PUSH    D       ;SAVE INDEX
        CALL    GETADR  ;GET A HEX NUMBER FROM KEYBOARD
        XCHG            ;RELOCATE NUMBER TO LESS CONVENIENT REGISTER
        POP     H       ;GET INDEX BACK
        MOV     A,L     ;MOVE TO ACCUMULATOR
        ANA     A       ;SET FLAGS
        JNZ     LOA030  ;IS IT THE PC REGISTER?
        PUSH    H       ;YES,
        LXI     B,PC    ;GET ADDRESS OF USER'S PC REG
        DAD     B       ;THIS IS REDUNDANT
        MOV     M,E     ;STORE DATA TO USER'S PC REGISTER
        INX     H       ;
        MOV     M,D     ;
        POP     H       ;
LOA030: LXI     B,PC1   ;STORE DATE TO USER REGISTER
        DAD     B       ;
        MOV     M,E     ;
        INX     H       ;
        MOV     M,D     ;
        CALL    BMPLNE  ;
        RET             ;
LDREGS: DB      'AHDBSP'        ;POSSIBLE REGISTER DESIGNATIONS
;
;STORE DATA TO MEMORY ROUTINE
;
MEMSTR: CALL    GETADR  ;GET BEGINNING ADDRESS
MEM010: PUSH    H       ;SAVE IT
        LHLD    CPOSIT  ;GET CURSOR POSITION
        MVI     A,'-'   ;DISPLAY PROMPT
        MOV     M,A     ;
        INX     H       ;UPDATE CURSOR
        SHLD    CPOSIT  ;RESTORE UPDATED CURSOR
        CALL    GETBYT  ;GET A DATA BYTE
        MOV     A,E     ;GET NUMBER OF BYTES RECIEVED
        CPI     2       ;CHECK FOR NONE RECIEVED
        JNZ     MEM020  ;ANY DATA RECIEVED?
        POP     H       ;NO, RESTORE HL
        RET             ;GET OUT OF HERE
MEM020: MOV     A,L     ;MOVE DATA BYTE TO ACCUMULATOR
        POP     H       ;GET ADDRESS BACK
        MOV     M,A     ;STORE DATA TO MEMORY
        INX     H       ;UPDATE ADDRESS
        JMP     MEM010  ;CONTINUE
;
;CALL USER SUBROUTINE
;
CALSUB: LHLD    CPOSIT  ;GET CURSOE POSITION
        MVI     A,'-'   ;DISPLAY PROMPT
        MOV     M,A     ;
        INX     H       ;UPDATE CURSOR POSITION
        SHLD    CPOSIT  ;RESTORE CURSOR
        MVI     A,0CDH  ;LOAD ACCUMULATOR WITH A CALL INSTRUCTION
        STA     INSTR   ;MOVE TO EXECUTION AREA
        CALL    GETADR  ;GET ADDRESS OF SUBROUTINE TO BE EXECUTED
        SHLD    INSTR+1 ;STORE ADDRESS TO EXECUTION AREA PLUS ONE
        CALL    BMPLNE  ;SCROLL UP
        CALL    OPEXEC  ;EXECUTE SUBROUTINE
        CALL    CLRSCR  ;CLEAR ANY GARBAGE FROM SCREEN
        CALL    SETSCR  ;SET UPSCREEN AGAIN
        RET             ;
;
;HIGH ORDER INSTRUCTION DECODE ROUTINE
;
OPHIGH: MOV     E,A     ;SAVE INSTRUCTION
        ANI     07H     ;DETERMINE SUB CLASS
        RLC             ;SCALE FOR INDEX
        MOV     C,A     ;POSITION FOR INDEXING
        MVI     B,0     ;
        LXI     H,HOPTAB        ;GET HIGH ORDER INSTRUCTION TABLE BASE A
        DAD     B       ;ADD INDEX
        MOV     A,E     ;RESTORE INSTRUCTION
        MOV     E,M     ;GET ADDRESS OF INSTRUCTION ROUTINE
        INX     H       ;
        MOV     D,M     ;
        XCHG            ;MOVE TO HL
        PCHL            ;PASS CONTROL TO APPROPIATE ROUTINE
;
HOPTAB: DW      RETURN  ;RETURN
        DW      PPINST    ;POP
        DW      JUMP    ;JUMP
        DW      MISC    ;MISC
        DW      CLINST    ;CALL
        DW      PSINST    ;PUSH
        DW      IMMEDT  ;IMMEDIATE
        DW      RESTRT  ;RESTART
;
HIGHOP: DB      'R   '  ;CONDITIONAL RETURN
HIHOP1: DB      'POP '  ;POP
        DB      'RET '  ;UNCONDITIONAL ROUTINE
        DB      '*NOP'  ;UNIMPLEMENTED OP CODES
        DB      'PCHL'  ;INDIRECT JUMP
        DB      'SPHL'  ;LOAD STACK POINTER WITH HL
HIHOP2: DB      'J   '  ;CONDITIONAL JUMP
HIHOP3: DB      'JMP '  ;UNCONDITIONAL JUMP
        DB      '*NOP'  ;UNIMPLEMENTED OP CODE
        DB      'OUT '  ;OUTPUT
        DB      'IN  '  ;INPUT
        DB      'XTHL'  ;EXCHANGE HL AND TOP OF STACK
        DB      'XCHG'  ;EXCHANGE HL AND DE
        DB      'DI  '  ;DISABLE INTERRUPTS
        DB      'EI  '  ;ENABLE INTERRUPTS
HIHOP4: DB      'C   '  ;CONDITIONAL CALL
HIHOP5: DB      'PUSH'  ;PUSH
        DB      'CALL'  ;UNCONDITIONAL CALL
        DB      '*NOP'  ;UNIMPLEMENTED OP CODE
HIHOP6: DB      'ADI '  ;ADD IMMEDIATE
```

Dynatrace, cont'd...

```
        DB      'ACI '  ;ADD IMMEDIATE WITH CARRY
        DB      'SUI '  ;SUBTRACT IMMEDIATE
        DB      'SBI '  ;SUBTRACT IMMEDIATE WITH BORROW
        DB      'ANI '  ;AND IMMEDIATE
        DB      'XRI '  ;EXCLUSIVE OR IMMEDIATE
        DB      'ORI '  ;OR IMMEDIATE
        DB      'CPI '  ;COMPARE IMMEDIATE
HIHOP7: DB      'RST '  ;RESTART
;
;DETERMINE CONDITIONAL FLAG ROUTINE
;
GTFLAG: ANI     38H     ;ISOLATE CONDITION BITS
        PUSH    H       ;SAVE HL
        RRC             ;SCALE FOR INDEXING
        RRC             ;
        MVI     D,0     ;
        MOV     E,A     ;CONSTRUCT INDEX
        LXI     H,FLAGS ;GET BASE ADDRESS OF FLAG CODES
        DAD     D       ;ADD INDEX
        MOV     A,M     ;GET FLAG CODE
        STAX    B       ;STORE TO VDM BUFFER
        INX     B       ;UPDATE POINTER
        INX     H       ;UPDATE FLAG LOCATOR
        MOV     A,M     ;GET SECOND CHARACTER OF FLAG
        STAX    B       ;DISPLAY IT
        LXI     D,'  '  ;OTHER CHARACTERS ARE BLANK
        POP     B       ;GET OLD HL
        CALL    MVNMNC  ;DISPLAY OP CODE
        RET             ;
;
;TEST FLAG FOR CONDITION
;
TSTFLG: ANI     38H     ;ISOLATE FLAG
        ORI     0C0H    ;ASSEMBLE A RETURN ON CONDITION INSTRUCTION
        STA     TST010  ;STORE FOR LATER EXECUTION
        LHLD    STSWRD  ;GET USER STATUS WORD
        PUSH    H       ;MOVE TO REAL FLAGS
        POP     PSW     ;
        MVI     A,1     ;A=1 MEANS CONDITION TRUE
TST010: RET             ;THIS IS REPLACED BY A CONDITIONAL RETURN
        MVI     A,0     ;CLEAR A IF CONDITION FALSE
        RET             ;
;
;CONDITIONAL JUMPS
;
JUMP:   LHLD    PC      ;GET USER PROGRAM COUNTER
        PUSH    H       ;SAVE IT
        PUSH    PSW     ;SAVE PSW
        MVI     B,3     ;THIS IS 3 BYTES
        CALL    MVINST  ;MOVE INSTRUCTION TO EXECUTION AREA
        POP     PSW     ;GET PSW BACK
        PUSH    PSW     ;SAVE IT AGAIN
        LXI     H,HIHOP2        ;ADDRESS OF MNEUMONIC
        LXI     B,HIHOP2+1      ;ADDRESS TO PUT FLAG INTO
        CALL    GTFLAG          ;WHAT CONDITION?
        POP     PSW             ;GET INSTRUCTION BACK
        CALL    TSTFLG          ;TEST CONDITION
        POP     H               ;GET PC AGAIN
        ANA     A               ;CHECK RESULT OF FLAG TEST
        RZ              ;RETURN IF RESULT WAS NEGATIVE
        INX     H       ;UPDATE PC
        MOV     A,M     ;GET ADDRESS TO JUMP TO
        STA     PC      ;AND STORE TO USER'S PROGRAM COUNTER
        INX     H       ;
        MOV     A,M     ;
        STA     PC+1    ;
        RET             ;
;
;THIS ROUTINE PROCESSES ALL IMMEDIATE MODE COMMANDS
;
IMMEDT: MOV     C,A     ;SAVE INSTRUCTION
        MVI     B,2     ;ALL ARE 2 BYTES
        CALL    MVINST  ;MOVE INSTRUCTION TO EXECUTION AREA
        MOV     A,C     ;RSTORE INSTRUCTION
        ANI     38H     ;DETERMINE TYPE OF IMMEDIATE INSTRUCTION
        RRC             ;SCALE FOR INDEXING
        MOV     E,A     ;POSITION FOR INDEXING
        MVI     D,0     ;
        LXI     H,HIHOP6        ;GET BASE ADDRESS OF IMMEDIATE CODES
        DAD     D       ;ADD INDEX
        MOV     B,H     ;POSITION FOR SUBROUTINE CALL
        MOV     C,L     ;
        LXI     D,'  '  ;OTHER CHARACTERS ARE BLANK
        CALL    MVNMNC  ;DISPLAY MNEUMONIC
        CALL    OPEXEC  ;EXECUTE INSTRUCTION
        RET             ;
;
;MISCELANEOUS INSTRUCTIONS
;
MISC:   PUSH    PSW     ;SAVE INSTRUCTION
        ANI     38H     ;DETERMINE TYPE
        RRC             ;
        MOV     E,A     ;
        MVI     D,0     ;
        LXI     H,HIHOP3        ;
        DAD     D
        MOV     C,L     ;
        MOV     B,H     ;
        LXI     D,'  '  ;
        CALL    MVNMNC  ;DISPLAY CODE
        POP     PSW     ;GET INSTRUCTION
        CPI     0E3H    ;
        JNC     XSDE    ;
        CPI     0D3H    ;
        JNC     IO      ;
        CPI     0CBH    ;
        JNC     NOOP    ;
        MVI     B,3     ;
        CALL    MVINST  ;
        LHLD    PC      ;
        DCX     H       ;
        MOV     A,M     ;
        STA     PC+1    ;
        DCX     H       ;
        MOV     A,M     ;
        STA     PC      ;
        RET             ;
NOOP:   LXI     B,HIHOP5+8
        LXI     D,'  '  ;
        CALL    MVNMNC  ;
        MVI     B,1     ;
        CALL    MVINST  ;
        RET             ;
IO:     MVI     B,2     ;
        CALL    MVINST  ;
        CALL    OPEXEC  ;
        RET             ;
XSDE:   MVI     B,1     ;
        CALL    MVINST  ;
        CALL    OPEXEC  ;
        RET             ;
;
;CONDITIONAL CALL ROUTINE
;
CLINST: PUSH    PSW     ;
        MVI     B,3     ;
        CALL    MVINST  ;
        POP     PSW     ;
        PUSH    PSW     ;
        LXI     H,HIHOP4        ;
        LXI     B,HIHOP4+1      ;
        CALL    GTFLAG  ;
        POP     PSW     ;
        CALL    TSTFLG  ;
        ANA     A       ;
        RZ
CAL010: LHLD    PC      ;
        XCHG            ;
        LHLD    STKPTR      ;
        DCX     H       ;
        MOV     M,D     ;
        DCX     H       ;
        MOV     M,E     ;
        SHLD    STKPTR      ;
        LHLD    PC      ;
        DCX     H       ;
        MOV     A,M     ;
        STA     PC+1    ;
        DCX     H       ;
        MOV     A,M     ;
        STA     PC      ;
        RET             ;
;
;CONDITIONAL RETURN
;
RETURN: PUSH    PSW     ;
        MVI     B,1     ;
        CALL    MVINST  ;
        POP     PSW     ;
        PUSH    PSW     ;
        LXI     H,HIGHOP        ;
        LXI     B,HIGHOP+1      ;
        CALL    GTFLAG  ;
        POP     PSW     ;
        CALL    TSTFLG ·;
        ANA     A       ;
        RZ              ;
RET010: LHLD    STKPTR      ;
        MOV     A,M     ;
        STA     PC      ;
        INX     H       ;
        MOV     A,M     ;
        STA     PC+1    ;
        INX     H       ;
        SHLD    STKPTR      ;
        RET             ;
;
;MISCELAEOUS STACK INSRUCTIONS
;
STKWRD: DB      ' B   D   H   PSW'      ;
;
PPINST: PUSH    PSW     ;
        ANI     08H     ;
        JNZ     NOTPOP  ;
        LXI     B,HIHOP1        ;
POP010: LXI     D,4     ;
        LXI     H,INSTA ;
        CALL    SETLNE  ;
        POP     PSW     ;
        PUSH    PSW     ;
        ANI     '0'     ;
        RRC             ;
        RRC             ;
        MOV     C,A     ;
        MVI     B,0     ;
        LXI     H,STKWRD        ;
        DAD     B       ;
        MOV     B,H     ;
        MOV     C,L     ;
        LXI     H,INSTA+4       ;
        LXI     D,4     ;
        CALL    SETLNE  ;
        POP     PSW     ;
        MVI     B,1     ;
        CALL    MVINST  ;
        CALL    OPEXEC  ;
        RET             ;
NOTPOP: POP     PSW     ;
        CPI     0C9H    ;
        JNZ     NOTRET  ;
        MVI     B,1     ;
        CALL    MVINST  ;
        LXI     B,HIHOP1+4      ;
        LXI     D,'  '  ;
        CALL    MVNMNC  ;
        JMP     RET010  ;
NOTRET: CPI     0F9H    ;
        JNZ     NTSPHL  ;
        MVI     B,1     ;
        CALL    MVINST  ;
        LXI     B,HIHOP1+16     ;
        LXI     D,'  '  ;
        CALL    MVNMNC  ;
        LHLD    HL      ;
        SHLD    STKPTR      ;
        RET             ;
NTSPHL: CPI     0E9H    ;
        JNZ     NOOP    ;
        MVI     B,1     ;
        CALL    MVINST  ;
        LXI     B,HIHOP1+12     ;
        LXI     D,'  '  ;
        CALL    MVNMNC  ;
        LXI     H,HL    ;
        MOV     A,M     ;
        STA     PC      ;
        INX     H       ;
        MOV     A,M     ;
        STA     PC+1    ;
        RET             ;
;
;PUSH AND UNCONDITIONAL CALL
;
PSINST: PUSH    PSW     ;
        ANI     08H     ;
        JNZ     CALLUN  ;
        LXI     B,HIHOP5        ;
        JMP     POP010  ;
CALLUN: POP     PSW     ;
        CPI     0CDH    ;
        JNZ     NOOP    ;
        LHLD    PC      ;
        PUSH    H       ;
        MVI     B,3     ;
        CALL    MVINST  ;
        LXI     B,HIHOP5+4      ;
        LXI     D,'  '  ;
        LXI     H,INSTA ;
        CALL    MVNMNC  ;
        POP     B       ;
        JMP     CAL010  ;
;
;RESTARTS
;
RESTRT: PUSH    PSW     ;
        MVI     B,1     ;
        POP     PSW     ;
        PUSH    PSW     ;
        ANI     38H     ;
        RRC             ;
        RRC             ;
        RRC             ;
        CALL    HEXASC  ;
        MOV     D,C     ;
        MVI     E,' '   ;
        LXI     B,HIHOP7        ;
        CALL    MVNMNC  ;
        LHLD    PC      ;
        XCHG            ;
        LHLD    STKPTR      ;
        DCX     H       ;
        MOV     M,D     ;
        DCX     H       ;
        MOV     M,E     ;
        SHLD    STKPTR      ;
        POP     PSW     ;
        ANI     38H     ;
        MOV     L,A     ;
        MVI     H,0     ;
        SHLD    PC      ;
        RET             ;
;
;THE GO ROUTINE CONTROLS SIMULATED EXECUTION OF USER PROGRAMS
;
GO:     LHLD    CPOSIT  ;
        MVI     A,'-'   ;
        MOV     M,A     ;
        INX     H       ;
        SHLD    CPOSIT  ;
        CALL    GETADR  ;
        SHLD    PC      ;
        CALL    BMPLNE  ;
        LXI     B,STPADR        ;
        LXI     D,8     ;
        LXI     H,LINE15        ;
        CALL    SETLNE  ;
        LXI     H,LINE15+8      ;
        SHLD    CPOSIT  ;
        CALL    GETADR  ;
        CALL    BMPLNE  ;
        XCHG            ;
GO010:  LHLD    PC      ;
        MOV     A,D     ;
        SUB     H       ;
        JNZ     GO020   ;
        MOV     A,E     ;
        SUB     L       ;
        RZ              ;
GO020:  PUSH    D       ;
        PUSH    H       ;
        LHLD    INSTSP  ;
GO030:  DCX     H       ;
        MVI     D,10H   ;
GO040:  DCR     D       ;
        JNZ     GO040   ;
        MOV     A,H     ;
        ANA     A       ;
        JNZ     GO030   ;
        CALL    STEP    ;
        CALL    DSREGS  ;
        POP     H       ;
        POP     D       ;
        IN      KSTAT  ;
        ANI     KBDRDY  ;
        RZ      ;REPLACE WITH RNZ FOR BOARDS WITH INVERTED I/O STATUS
        JMP     GO010   ;
STPADR: DB      'STOP AT-'      ;
;
;THIS ROUTINE READS INTEL FORMAT TAPES
;
READTP: CALL    BMPLNE  ;
        CALL    TAPEIN  ;
        MOV     A,C     ;
        RZ              ;
        MVI     B,0     ;
        LXI     D,16    ;
        SUI     10H     ;
        MOV     C,A     ;
        LXI     H,ERRMES        ;
        DAD     B       ;
        MOV     B,H     ;
        MOV     C,L     ;
```

```
        LHLD    CPOSIT  ;
        CALL    BMPLNE  ;
        CALL    SETLNE  ;
        CALL    BMPLNE  ;
        RET             ;
;
;
;
;
TAPEIN: MVI     C,0     ;
        MVI     A,0FFH  ;
        OUT     SWTCHS  ;
BBLK:   CALL    INPUT   ;
        CPI     3AH     ;
        JNZ     BBLK    ;
        MVI     B,0     ;
        CALL    RDBYTE  ;
        MOV     D,A     ;
        CALL    RDBYTE  ;
        MOV     H,A     ;
        CALL    RDBYTE  ;
        MOV     L,A     ;
        CALL    RDBYTE  ;
        MOV     E,D     ;
        INR     E       ;
LD10:   DCR     E       ;
        JZ      LD20    ;
        MOV     A,L     ;
        CMA             ;
        OUT     SWTCHS  ;
        CALL    RDBYTE  ;
        MOV     M,A     ;
        CMP     M       ;
        INX     H       ;
        JZ      LD10    ;
        MVI     C,20H   ;
        RET             ;
LD20:   MOV     A,H     ;
        CMA             ;
        OUT     SWTCHS  ;
        CALL    RDBYTE  ;
        MOV     A,B     ;
        ORA     A       ;
        JZ      NBLK    ;
        MVI     C,10H   ;
        RET             ;
NBLK:   MOV     A,D     ;
        ORA     A       ;
        JNZ     BBLK    ;
        RET             ;
;
;
;
RDBYTE: PUSH    D       ;SAVE LENGTH
        CALL    INDIGT  ;GET 1 HEX DIGIT FROM TAPE
        ADD     A       ;MULTIPLY BY 16
        ADD     A       ;
        ADD     A       ;
        ADD     A       ;
        MOV     D,A     ;SAVE MSD
        CALL    INDIGT  ;GET NEXT HEX DIGIT FROM TAPE
        ORA     D       ;COMBINE HEX DIGITS TO FORM BYTE
        MOV     D,A     ;SAVE BYTE WHILE DOING CHKSUM
        ADD     B       ;ADD CHKSUM TO THE NEW BYTE
        MOV     B,A     ;REPLACE OLD CHKSUM WITH NEW
        MOV     A,D     ;GET NEW BYTE BACK
        POP     D       ;RESTORE LENGTH
        RET
;
;
;
INDIGT: CALL    INPUT   ;READ A FRAME FROM TAPE
        CPI     '9'+1   ;INSPECT DATA
        JM      IND010  ;OK IF DATA < 10
        ADI     9       ;ELSE ADJUST FOR ASCII BIAS
IND010: ANI     0FH     ;REDUCE MOD 16
        RET             ;
;
;
INPUT:  IN      RSTAT   ;GET READER STATUS FROM 3P+S
        ANI     RDRRDY  ;TURN OFF NON READER BITS
        JNZ     INPUT   ;LOOP UNTIL DATA AVAILABLE
        IN      READER  ;GET DATA
        ANI     7FH     ;CLEAR PARITY BIT
        RET             ;
;
;TAPE READER ROUTINE ERROR MESSAGES
;
ERRMES: DB      'CHECK SUM ERROR '
        DB      'MEMORY FAILURE  '
;
;THIS ROUTINE ZEROES A BLOCK OF MEMORY
;
ZERMEM: CALL    GTLMTS  ;
ZER010: MVI     A,0     ;
        MOV     M,A     ;
        INX     H       ;
        MOV     A,D     ;
        SUB H           ;
        JNZ     ZER010  ;
        MOV     A,E     ;
        SUB     L       ;
        JNZ     ZER010  ;
        RET             ;
;
;
;
ISPEED: CALL    KEYBDI  ;
        CALL    ASCHEX  ;
        ADI     01H     ;
        MOV     H,A     ;
        MVI     L,0     ;
        MVI     A,10H   ;
        SUB     H       ;
        MOV     H,A     ;
        SHLD    INSTSP  ;
        CALL    BMPLNE  ;
        RET             ;
INSTSP: DW      0400H   ;
;
;
;
        END
```

THE CP/M CONNECTION

Chris Terry

Part 1 - Interfacing To the Operating System: Relocating CP/M

The CP/M system requires at least 16K of contiguous RAM for a minimal system. Page 0 is always reserved for entry points, file control blocks, and a 128-byte buffer used for command input from the console and as the default disk input/output buffer. Other buffer locations can be specified by an application program with a function call to BDOS.

The CP/M system proper is always located at the top of the available memory; in the minimal 16K system distributed by most disk controller manufacturers, the CCP starts at 2900H and the CBIOS at 3E00H. When more memory becomes available, the system can be relocated to the top of the new memory, so as to leave more room in the Transient Program Area (TPA) for application programs and data. This feature makes CP/M extremely versatile, because the addressable memory area above the RAM block can be used for PROM containing I/O routines and utilities without conflicting in any way with the CP/M requirements.

Four CP/M utilities are required for creating a relocated system and putting it on a fresh diskette:

```
MOVCPM
ASM
DDT
SYSGEN
```

Obviously, moving the CCP, BDOS, and CBIOS to a new location requires that all of the CALL and JMP addresses be changed to fall within the new system area. Equally obviously, we cannot change anything in the system that is currently up and running or it would crash. Therefore, to relocate the system, we use the MOVCPM utility, which contains a complete set of the system machine code. If we wish to create a new 32K CP/M system, we invoke MOVCPM with the command:

```
A>MOVCPM 32 *
```

MOVCPM now changes all the CALL and JMP addresses in its internal version of CP/M to suit the size we have requested (in this case, 32K), and then places the reconstructed CP/M code in the TPA with the Boot (Cold Start Loader) starting at 900H, the CCP starting at 980H, and the BIOS starting at 1E80H. Now it tells us that the new system is ready for SYSGEN or for the command:

```
SAVE 32 CPM32.COM
```

and DDT will automatically put the reconstructed CP/M at the right place (980H).

Let's look at figure 1. We see that our Boot has to be loaded at 900H. The execution addresses in BOOT.HEX start at 0000, and if we just used the simple Read (R) command of DDT, that is where the Boot would be loaded. However, DDT allows us to use an OFFSET with the read (R) command; this offset is added to every load address. We calculate it by taking the difference between the address where we want loading to start and the ORG address of the file. If BOOT is ORGed at 0000, the offset is 0900H-0000=900H; if BOOT is ORGed at 80H. the offset is 0900-0080H=880H. If we don't have a hex calculator (such as the TI Programmer), we can use the hex arithmetic command (H) of DDT; the command:

```
-H900,80
```

will cause DDT to give us first the sum (980) and then difference (880) of our two numbers. So, to overlay the MDS Boot with our own, we give the commands:

```
-IBOOT.HEX
-R900 (if ORG is 0000)
or
-R880 (if ORG is 80H)
```

Things become a little more tricky when we come to the CBIOS. Look at figure 1 for a moment. Moving the Boot to the Memory Image area was simple, because we were moving it upward. But to get the CBIOS shifted from its execution address of 7E00 to 1E80 in the Memory Image area, we have to shift it DOWNWARD. Unfortunately, DDT can only ADD an offset to, not subtract it from, the file load address. However, we can still use a positive offset that will bring us to the right place, because the CPU address counter has only 16 bits; thus, if we add 1 to address FFFF we get 10000 -- but the counter has no place to put the leftmost digit, so we come back to 0000. We see, then, that to shift a program downward in memory, we have to give DDT an offset that will push the program up off the top of memory and bring it upward through the bottom. A Two's Complement subtraction of the larger address from the smaller will do precisely this.

We know that for a 16K system the offset is 980-2900=E080, and since 3E00 is the execution

Chris Terry, 324 E. 35th St., New York, NY 10016

CP/M Connection, cont'd...

address of CBIOS, it will be found in the Memory Image area at:

3E00 + E080 = 11E80 = 1E80

We also know that the start of the CCP in our new system is 2900+4000=6900, so the offset for our new system is 980-6900=A080. If we add this to the start of our new CBIOS, which is 3E00+BIAS=7e00, we find that 7E00+A080=1E80. Bingo! Now, to overlay the MDS BIOS (which was put in the Memory Image area at 1E80 by MOVCPM) we tell DDT:

```
-ICBIOS.HEX
-RA080
```

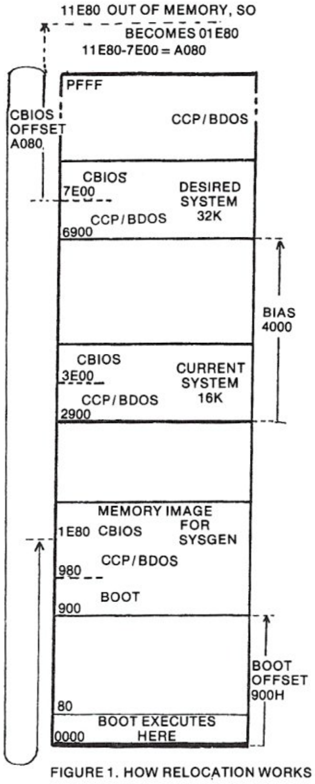


FIGURE 1. HOW RELOCATION WORKS

We can forget about SYSGEN at this point, because the Boot and the BIOS generated by MOVCPM are for the Intel MDS system and chances are that not even the console I/O would work, let alone the disk commands. So we save the COM file, as instructed.

Now we have some preparation work to do, so that we can overlay the MDS Boot and BIOS with the Boot and CBIOS supplied by our controller manufacturer and previously modified to work with our system I/O. First, we must calculate the BIAS for our new system; that is, the amount by which every CP/M instruction is shifted upward. MOVCPM took care of this for the CCP and BDOS instructions, but we need it to find where to ORG our own CBIOS. We are going to move the system up by 16K; since 4096 decimal (4K) is equivalent to 1000 hex, it follows that our BIAS for a 32K system is going to be 4x1000=4000H. We have to apply this BIAS to three items:

In the Coldstart Loader, to set:
    (1) the address at which CP/M will be loaded, and
    (2) the address to which the loader will jump to start CP/M running;
In the CBIOS, to set:
    (3) the ORG address.

Let's look at the ASM listing of the Boot (using our editor), and find out where it executes. It will almost certainly be either 0000 (as in the case of the Tarbell disc controller board) or 80H (as in the case of the Thinker Toy board). We make a note of the ORG address, but do not change it. If we find an MSIZE EQU 16 statement, this will be used by the assembler to compute the load address (1) and the jump address (2); we need only change it to MSIZE EQU 32. If we do not find an MSIZE equate, the 4th executable statement will be LXI H,2900H; we change the operand to LXI H,6900 (=2900+BIAS). The 3rd statement of the RBLK1 routine will be JZ 3E00H; we change this to JZ 7E00H (=3E00+BIAS).

Next, we must go into the CBIOS.ASM file with our editor, and look at the ORG statement. Once again, if we find an MSIZE equate statement, that is all we need to change. The Assembler will do the rest.

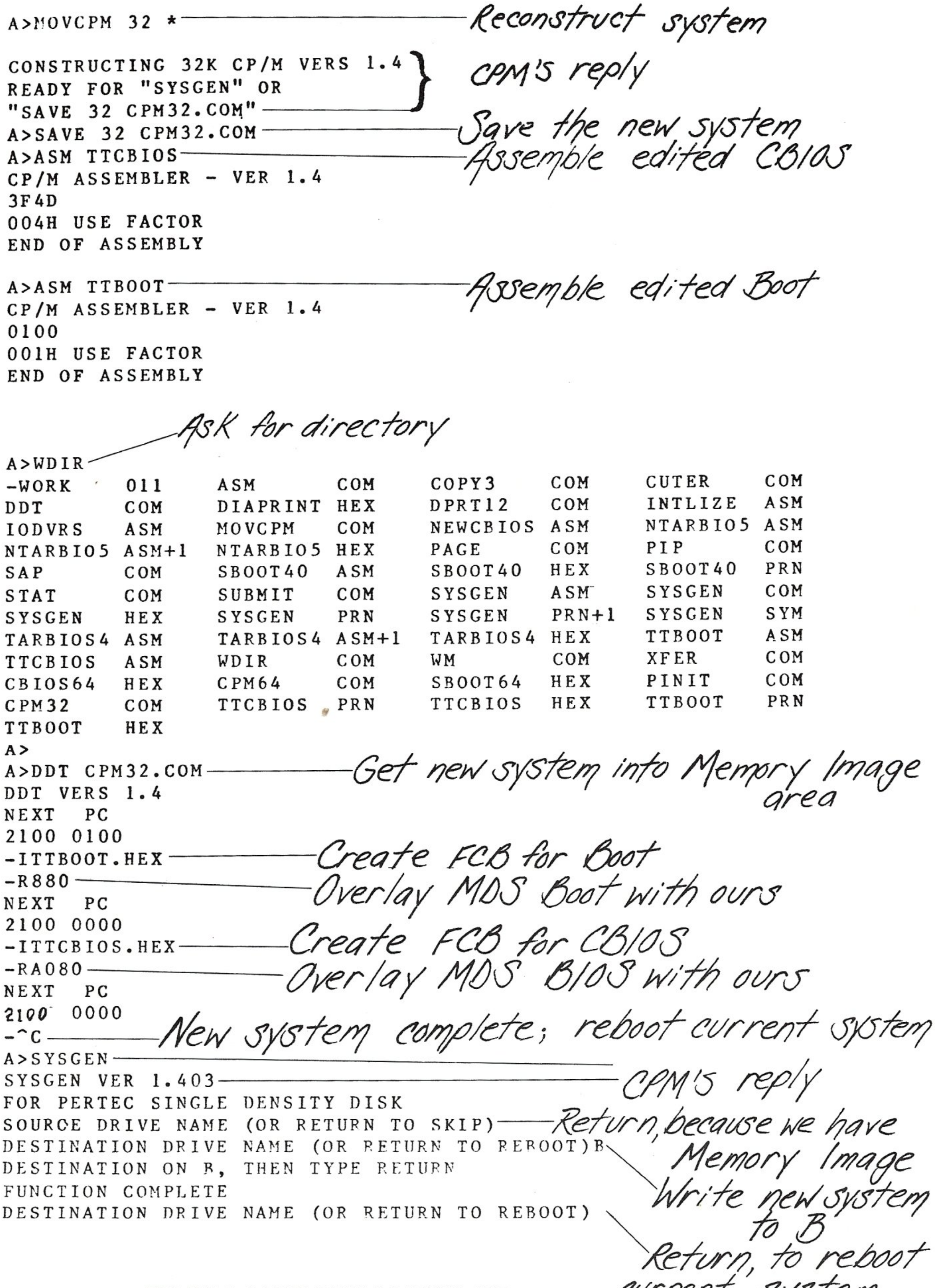
If we do NOT have an MSIZE statement, but instead find ORG 3E00H (or any other absolute address), we have to change the ORG address to 3E00+BIAS (in this case 7E00). Assuming that we don't want to make any changes to the peripheral drivers (for console, list, reader, or punch), we can exit from the editor.

Now we use ASM to reassemble the CBIOS at the new address. It's a good idea to let the assembler create a PRN listing with all the addresses and object code, and to print this right away. We can then ERAse CBIOS.PRN, which takes up a lot of room on the disk and is not required any more. Before going any further, let's check the directory and make sure that we do indeed have the CPM32.COM, CBIOS.HEX, and BOOT.HEX files.

CALCULATING OFFSET

At this point we have to remember that every COM and .HEX file contains built-in instructions on where to load it. We shall have no trouble with the CPM32.COM which we saved previously, because MOVCPM arranged for the reconstructed CCP and BDOS to be loaded into the Memory Image area starting at 980H, even though they will execute at 2900+BIAS (6900 in this case). We have only to give the command:

```
A>DDT CPM32.COM
```



```
A>MOVCPM 32 *                                   Reconstruct system

CONSTRUCTING 32K CP/M VERS 1.4  }               CPM'S reply
READY FOR "SYSGEN" OR           }
"SAVE 32 CPM32.COM"             }
A>SAVE 32 CPM32.COM                             Save the new system
A>ASM TTCBIOS                                   Assemble edited CBIOS
CP/M ASSEMBLER - VER 1.4
3F4D
004H USE FACTOR
END OF ASSEMBLY

A>ASM TTBOOT                                    Assemble edited Boot
CP/M ASSEMBLER - VER 1.4
0100
001H USE FACTOR
END OF ASSEMBLY

                 Ask for directory
A>WDIR
-WORK    011     ASM      COM      COPY3    COM      CUTER    COM
DDT      COM     DIAPRINT HEX      DPRT12   COM      INTLIZE  ASM
IODVRS   ASM     MOVCPM   COM      NEWCBIOS ASM      NTARBIO5 ASM
NTARBIO5 ASM+1   NTARBIO5 HEX      PAGE     COM      PIP      COM
SAP      COM     SBOOT40  ASM      SBOOT40  HEX      SBOOT40  PRN
STAT     COM     SUBMIT   COM      SYSGEN   ASM      SYSGEN   COM
SYSGEN   HEX     SYSGEN   PRN      SYSGEN   PRN+1    SYSGEN   SYM
TARBIOS4 ASM     TARBIOS4 ASM+1    TARBIOS4 HEX      TTBOOT   ASM
TTCBIOS  ASM     WDIR     COM      WM       COM      XFER     COM
CBIOS64  HEX     CPM64    COM      SBOOT64  HEX      PINIT    COM
CPM32    COM     TTCBIOS  PRN      TTCBIOS  HEX      TTBOOT   PRN
TTBOOT   HEX
A>
A>DDT CPM32.COM                     Get new system into Memory Image
DDT VERS 1.4                                                    area
NEXT  PC
2100 0100
-ITTBOOT.HEX                        Create FCB for Boot
-R880                               Overlay MDS Boot with ours
NEXT  PC
2100 0000
-ITTCBIOS.HEX                       Create FCB for CBIOS
-RA080                              Overlay MDS BIOS with ours
NEXT  PC
2100 0000
-^C                   New system complete; reboot current system
A>SYSGEN
SYSGEN VER 1.403                                        CPM'S reply
FOR PERTEC SINGLE DENSITY DISK
SOURCE DRIVE NAME (OR RETURN TO SKIP)        Return, because we have
DESTINATION DRIVE NAME (OR RETURN TO REBOOT)B       Memory Image
DESTINATION ON B, THEN TYPE RETURN                Write new system
FUNCTION COMPLETE                                         to B
DESTINATION DRIVE NAME (OR RETURN TO REBOOT)      Return, to reboot
                                                  current system
```

FIGURE 2. SAMPLE RELOCATION JOB

CP/M Connection, cont'd...

If it should be necessary at this stage to make minor changes to the CBIOS, it is now easy to fine the address at which the change is to be made. Any address in the Memory Image area can be found by adding the offset to the execution address shown in the CBIOS.PRN listing of the re-assembled CBIOS. We can then use either the DDT Substitute (S) command to insert the new hex values or, if several successive instructions are to be changed, we can use the DDT A (Assemble) command which allows complete instructions to be inserted using the Intel mnemonics for operation and register codes, and hex values for addresses or constants.

The CP/M System Alteration Manual has all this information; there is even a table of offsets for various system sizes. But for some reason I and many others have great difficulty in getting the procedure clear. Perhaps the CP/M manual gives us too much, too quickly -- that is why I have spreadout this description.

THE USES OF 'SYSGEN'

At this point, the Memory Image area contains a complete 32K CP/M system with the correct CBIOS and Boot for our own configuration. We now use the SYSGEN utility to write the new system out to Tracks 0 and 1 of a fresh, formatted disk. The complete sequence of commands is shown in figure 2, with comments.

Note that when SYSGEN asks SOURCE DRIVE NAME (OR RETURN TO SKIP), we hit Return because we already have the reconstructed system in the memory area. However, when we are not relocating CP/M, but merely putting the existing system, unchanged, onto a new disk, we do not need to use DDT to get the current system into the Memory Image area. When SYSGEN asks for the source drive, we tell it A, and CP/M is read from Tracks 0 and 1 of our current system disk into the Memory Image area. Then, when SYSGEN asks for the destination, we tell it B, and write the system out to a new disk.

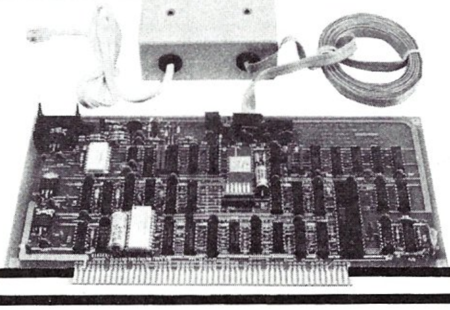
COPYING THE SYSTEM FILES

SYSGEN does not handle anything except the CP/M system itself. The utilities, such as ASM, ED, DUMP, LOAD, etc., must be handled separately. The files on our current system disk can be transferred over to the new system either by the CP/M Users' Group utility COPY.COM, or by the command

```
A>PIP B:=A:*.*[V]
```

COPY transfers a whole track at a time, verifying each sector but not reporting until the end. PIP transfers one file at a time, verifies the new file against the old, and reports the name of the file transferred, so it takes somewhat longer. PIP will be our choice if we want to be selective, copying only the .COM files, for example.

In the next part of this article I will discuss handling the I/O byte.



NorthStar Topics

A Spooling Program and Much More

Randy Reitz

The program described here is an extension to the North Star DOS which implements input/output redirection, a printer SPOOL and a command stack. The word SPOOL is an acronym for Simultaneous Printing of Output on Line. The word "simultaneous" is used in the sense of driving multiple printers. I have borrowed the acronym and also changed its meaning in this application. This program allows the redirection of all character input/output which use the N*DOS drivers (located at 2900H-29FFH in the single density DOS). This means that all character I/O can use the device specified and/or use a disk file. This feature provides a lot of flexibility for transferring data between programs which can typically be done only on those "big" systems. This program will also drive a printer from a disk file while the computer is waiting for terminal input, so I feel the "SPOOL" acronym applies.

Here is a list of the features of this program:

- —Full I/O redirection for a terminal, printer and disk files of type 7.
- —Multiple commands can be entered on one line using the "command stack."
- —An "operating system" is included which allows buffered or unbuffered access to any North Star disk file.
- —A significant amount of error checking is done (the exception is hard disk errors).

The program is 2.5K bytes long (the same size as the N*DOS) which includes buffer space for a read, write and printer spool file. This implementation is a stand alone program which can be located anywhere in memory (only slight adjustments are required to put the program in EPROM memory). The "operating system" provides entry points to open an existing file, to do blocked read and write, unblocked read and write (i.e. one byte-at-a-time access of disk files), to close and create a new file. These are capabilities found in operating systems like CP/M. This "operating system" uses file control blocks and provides for a pseudo-dynamic file system. If a new file is created with a length of zero, all available disk space is used. When this file is closed, the disk space is adjusted to the amount actually used.

I first started developing this program two years ago. It has evolved into it's present form which I feel is the easiest to implement. The only "personalization" required is to tell the program the address of the terminal status port and the bit which indicates when a character is ready (bit is set) on the keyboard. The N*DOS I/O drivers don't provide this function and this program needs to know how to detect when a character is ready on the keyboard.

The first part of this article will discuss I/O redirection, the command stack and the printer spool sections of the program. The second part will discuss the "operating system" and the error handling routines. This program will work with N*DOS release 4.3 or 5.1S. I do not have double density and I suspect that the changes required to accomodate double density would not be trivial.

The program begins with constant declarations. The only constant which needs to be changed for release 4.3 is DOSPTNR (change to 28FAH). All of the references to the DOS program, except the DOSPTNR constant, are documented in the N* System Software Manual. As I will soon discuss, output can be directed to a disk file while the DOS is requested to list the contents of the disk directory. I found that the DOS does not expect any other disk activity to be occuring while it is printing the directory. In order to request a DOS directory look up (using entry DLOOK) while the DOS already has a directory maintenance activity in progress, the 2-byte value at location DOSPTNR must be preserved. The N*DOS is not designed to be a reenterant program. The references to the MDS firmware assume a standard PROM is installed at E800H.

This program's origin is at 1600H and three 256-byte buffers follow starting at 1DOOH. These addresses place the entire program and buffers in the 2.5K bytes immediately below the DOS. These addresses can be changed to place the program and buffers anywhere in memory. They don't have to be in contiguous memory locations. The only consideration required to locate the program in EPROM is to move the storage area at the end of the program to a suitable memory address as well as the jump vectors for the DOS I/O routines. If the program is named "DOSPOOL", then the N*DOS command "GO DOSPOOL" will load the program and jump to 1600H. The program begins with a jump over the constants needed to determine if a character is ready on the keyboard and the hold area for the DOS I/O jump vectors.

This program begins by inserting itself between the DOS user programs and the DOS I/O routines. The initialization routine at DOSPOOL changes the addresses in the DOS to jump to this program's character input and output routines. The addresses currently used by the DOS are stored in the JMP instructions labeled USRCOUT and USRCIN. The addresses for the command stack are then initialized and finally the terminal status port and keyboard mask are written into the program. If you implement the program in EPROM, you should insert these two bytes before "burning" the

Randy Reitz, 26 Maple St., Chatham Township, NJ 07928

EPROMS. As I mentioned before, the program needs to determine when a character is ready on the terminal keyboard. This is required when the program is driving the printer from a disk file while waiting for terminal input. The values given are for the standard HORIZON serial port. The initialization is now finished so the DOSPOOL routine returns to the DOS. The DOS will immediately enter at SPLCIN by requesting terminal input.

The program continues with the routines used for character input/output. The I/O redirection is controlled by the IOFLAG byte which is similar to the IOBYTE used in CP/M. If the IOFLAG byte is 0, all I/O uses the device specified by the value of the A-reg when the input/output routine is entered. The individual bits in IOFLAG are set using escape sequences (this is a two character sequence, the first character of which is an escape - ASCII 27).

The SPLCIN routine is entered by the DOS whenever the DOS or a user program (e.g. N*BASIC) requests a character from the device specified in the A-req. Since the routines which redirect I/O to the disk use a lot of stack space, the SPLCIN (and SPLCOUT) routine reset the machine stack to the area provided at address IOSTK. The main character input routine FETCH is called by SPLCIN. When FETCH returns with bit 7 reset, SPLCIN returns to the calling program with the character in the A-reg and all registers restored.

The FETCH routine is long because it recognizes the escape sequences which control the value of the IOFLAG byte and the names of the spool files. Since the command stack is given the highest priority, FETCH begins by checking if the command stack is active. If so, the next character is taken from the stack and returned to the calling program. The command stack is used by separating commands on a line by the logical end-of-line character (I chose "I"-ASCII 124). The escape and "I" characters are special to the FETCH routine. If you have to return one of these characters to the calling program, you need to type them twice. A typical example which uses the command stack is:

CR STUFF 23/TY STUFF 2 <CR>

By typing this line, the DOS program would receive the CR STUFF 23 part of the line normally when the carriage return (<CR>) key is typed at the end of the line. During typing, the "I" would stop sending characters to the calling program and enter all remaining characters typed into the command stack. The <CR> key typed at the end of the line is sent to the calling program. When the calling program returns for more characters, all characters up to the next "I" or the actual end-of-line are returned to the program along with another <CR>. I use this feature to stack commands so my disk drive motors do not shut-off due to my slow typing. Another feature of the command stack is that the escape sequence ESC S will cause the FETCH routine to start sending characters from the current contents of the command stack. If you have one command or a series of commands you want to execute repeatedly, you can put them in the stack by starting a line with a "I" character and then type the command (or commands separated by "I" characters) followed by a <CR> character. The <CR> will be sent to the calling program; but then typing an ESC S will send these command(s) to the calling program whenever desired.

After the command stack is checked, the next check in the FETCH routine is to see if the read disk spool is active. The appropriate bits in the IOFLAG are checked, and if active, all input is read from the read spool file. This will continue until the read spool file reaches an end-of-file condition or any character is typed on the terminal keyboard. Both of these conditions will stop the read spool until the IOFLAG byte is cleared. One character is read from the read spool file whenever a device code of 8 is used. The IOFLAG byte contains a bit which will cause the input device code of 8 to be ignored and passed to the user's CIN routine.

At the point KEYBD in the FETCH routine, neither the command stack or input spool file is active, so a character is obtained from the requested input device. The GETCP routine is used which will drive the printer with the contents of a printer spool file while waiting for the user to type a character. Once GETCP returns with a character, the special characters of ESC and "I" are checked. If these characters are not found, the character typed is returned to the calling program. If a logical end-of-line character ("I") is found, control is transferred to LDSTK which will continue to read characters into the command stack. The remainder of the FETCH routine handles the escape sequences. Here is a complete list of escape sequences:

Char after ESC	**Function**
0	Zeroes the IOFLAG byte which clears any spool errors and stops the print spool.
1	Directs all output to both the printer (device 1) and the terminal (device 0).
2	Opens the print spool and begins printing, if the print spool was already open then printing resumes.
4	Directs all output to the terminal and write spool file.
8	Reads all input from the read spool file until end-of-file or any character is typed.
@	SPLCIN and SPLOUT will ignore the device code in the A-reg. SPLCIN will pass the device code to the user's CIN routine. SPLOUT will force a device code of 0.
R	Displays the current name of the read spool file. Type a <CR> if no change is desired or else type the name of the file. Use N*DOS file

North Star, cont'd...

Char after ESC	Function
	name conventions. Use a BS char to correct typing errors.
W	Same as R but for write spool file name.
O	Same as R but for print (output) spool file name.
^S	Start sending characters from current contents of command stack. Continues until end of stack.
^C	Close current write spool file and set file length to number of blocks used.
ESC	Send the ESC character to the calling program.
I	Send the "I" character to the calling program.

If the R, W and O commands are used to examine the name of a spool file only, the status of the file is not changed. Hence, the 0 and 4 commands can be used to append characters to the end of an existing write spool file that is open. If the name of a spool file is changed, the file is closed. Therefore, to restart a spool file from the beginning, merely retype its name. If a spool error occurs during a read or write spool file operation, the error bit in the IOFLAG is set and spooling stops. The 0 command will clear the IOFLAG and spooling can be resumed at the point of the error if possible. Finally, if a program is writing to device 4 (the write spool file), the program can close the write spool file by writing an SOH character (ASCII 1). Similarly, the read spool will terminate when an SOH character is found.

The SPLOUT routine is not as long as the SPLCIN routine. This routine also resets the machine's stack pointer and then checks if the write spool file is active in the IOFLAG. One problem in the SPLOUT routine is that when a program is reading from device 8 (the read spool file) it may "Echo" characters to device 8 also. I chose to ignore these echoes so that a read and write spool can be active simultaneously.

The next routines implement the command stack. The entry GOSTK will pull characters from the stack and the entry LDSTK will put characters from the stack and the entry LDSTK will put characters onto the stack. When loading characters onto the stack, the BS (backspace) character is used to delete the previous character typed. Also the characters entered on the stack are echoed to device 0. This may be a problem if some other input device uses the command stack.

The next routines are the spool drivers. Entry GETSPL will read one character from the read spool file. If the read spool file is not open, SPLSRCH is called to open the file. RDERR is used to report (on device 0) any error that occurs on the open. RDDISK will get the next character from the file. If the character is a CR , then CKEYBD is used to detect if a key has been hit on the keyboard (device 0). Typing any key while the read spool is active will stop the read spool. Spooling can be resumed by first clearing the IOFLAG byte with ESC 0. If the program is reading from device 8 then spooling will resume, else an ESC 8 will do it. The errors that are detected are bad filename, file not found, file type error and length error.

Entry PUTSPL will write a character to the write spool. If the write spool is not open, SPLSRCH is called to open the file. If the file doesn't exist, CREATE is called to create the file using as much of the disk space as is available. If an SOH character is recieved, the write spool file is closed. The errors that are detected are disk write protected, disk full, length error and bad filename.

Entry GETCP will drive a printer if bit 1 is set in the IOFLAG (ESC 2). If this bit is not set, control is transferred to GETCHAR which waits for a character to be typed, or else PRACT is called. This routine checks if the print spool file is open and opens it if required. This routine will loop at location PRLOOP while checking the keyboard with GETSTAT. PRLOOP expects to be driving a printer at 30 CPS. When there are 38 characters left in the buffer, STRTM is called to start the disk drive motors. When there are 8 characters left in the buffer, SELECT is called to load the disk drive head. These constants can be varied to suit your printer so that the printing will not stop while the disk drive turns on. This "nice" feature was really necessary since the N*DOS will loop for l-sec after turning on the disk drive motors. If this happens, no character typed on the keyboard would be found until the DOS returned. This would cause an unacceptable slowdown in typing. The errors that are detected are bad filename, file not found, file type error and length error.

The SPLSRCH and CLSPOOL routines are short interface routines to the "operating system." The next routines display and allow the names of the three spool files to be changed. A name is considered changed if any character other than a CR is typed. When the name of a spool file is changed, it is considered to be closed. Finally, some miscellaneous routines are given in the listing. The "operating system" will be discussed in the next part.

I find that I use this program most frequently to transfer data between programs. Although I began by implementing a spool file to drive a printer, the possibilities available by writing and reading spool files quickly became more important.

—PROGRAM BEGINS NEXT PAGE—

Introduction To CP/M

Part IV of the "Introduction To CP/M," by Jake Epstein, will be continued in the next issue of S-100 MICROSYSTEMS. Regretfully, the manuscript arrived too late for inclusion in this issue.

```
0000             0005 *ASCII CONSTANTS
0000             0010 CTRLC     EQU   'C'-40H
0000             0015 CTRLS     EQU   'S'-40H
0000             0020 BELL      EQU   7
0000             0025 ESC       EQU   1BH
0000             0030 LF        EQU   10        LINE FEED
0000             0035 BS        EQU   8
0000             0040 CR        EQU   13        CARRIAGE RET
0000             0045 RUBOUT    EQU   7FH
0000             0050 *
0000             0055 *EXTERNAL REFERENCES - DOS
0000             0060 COUT      EQU   200DH     CHARACTER OUT ROUTINE
0000             0065 CIN       EQU   2010H     CHARACTER IN ROUTINE
0000             0070 DLOOK     EQU   201CH     FILE DIRECTORY LOOKUP
0000             0075 DWRIT     EQU   201FH     WRITE A DIRECTORY BLOCK
0000             0080 DCOM      EQU   2022H     ISSUE A DISK COMMAND
0000             0082 DOSERR    EQU   202CH     FILE NAME OR DSK WP ERR
0000             0085 DOSPTNR   EQU   28B7H     POINTER ADDRESS IN REL 5.1
0000             0090 DOSENTR   EQU   2028H     DOS RE-ENTRY POINT
0000             0095 *
0000             0100 *EXTERNAL REFERENCES - MDS FIRMWARE
0000             0105 DRVSEL    EQU   2003H     CONTAINS CURRENT DRIVE SELECTED
0000             0110 READA     EQU   0EB10H    READ A STATUS
0000             0115 RESET     EQU   0EB90H    RESET MOTOR TIMER
0000             0120 NOSEL     EQU   59H       FLAG FOR NO DRIVE SELECTED
0000             0125 MO        EQU   10H       MOTOR-ON BIT
0000             0130 CC        EQU   0EBH      CONTROLLER COMMAND
0000             0135 *
0000             0140 *OTHER CONSTANTS
0000             0145 SPLPRG    EQU   1600H
0000             0150 SPLBUF    EQU   1D00H
0000             0155 *
0000             0160           ORG   SPLPRG
1600             0165 *
1600             0170 *THE "GO DOSPOOL" COMMAND ENTERS HERE
1600             0175 *
1600 C3 0B 16    0180           JMP   DOSPOOL
1603             0185 *
1603             0190 *HERE ARE THE CONSTANTS NEEDED TO DETERMINE
1603             0195 *IF A CHARACTER IS READY ON THE KEYBOARD
1603             0200 *
1603 03          0205 STATPRT   DB    3         THE "ADDRESS" OF THE STATUS PORT
1604 02          0210 KBDMSK    DB    2         THE KEYBOARD READY BIT
1605             0215 *
1605             0220 *HOLD THE JUMP VECTORS FOR DOS I/O ROUTINES
1605             0225 *
1605 C3 05 16    0235 USRCOUT   JMP   USRCOUT
1608 C3 08 16    0236 USRCIN    JMP   USRCIN
160B             0240 *
160B             0245 *INITIALIZE THE SPOOLING PROGRAM
160B             0250 *CHANGE THE CHARACTER I/O ADDRESSES IN DOS
160B             0255 *AND INITIALIZE THE PROGRAM CONSTANTS
160B             0260 *
160B 2A 11 20    0265 DOSPOOL:  LHLD  CIN+1     GET USER'S INPUT ROUTINE ADDR
160E 22 09 16    0270           SHLD  USRCIN+1
1611 2A 0E 20    0275           LHLD  COUT+1    GET USER'S OUTPUT ROUTINE ADDR
1614 22 06 16    0280           SHLD  USRCOUT+1
1617 21 44 16    0285           LXI   H,SPLCIN GET SPOOL INPUT ROUTINE ADDR
161A 22 11 20    0290           SHLD  CIN+1     SO DOS COMES HERE
161D 21 DF 16    0295           LXI   H,SPLCOUT  SAME FOR OUTPUT ROUTINE
1620 22 0E 20    0300           SHLD  COUT+1
1623 21 95 1C    0305           LXI   H,LOCSTK START OF COMMAND STACK
1626 22 52 1C    0310           SHLD  STACK
1629 21 00 00    0315           LXI   H,0
162C 22 50 1C    0320           SHLD  RDSTK     COMMAND STACK EMPTY
162F 22 54 1C    0325           SHLD  IOFLAG
1632 3E FF       0330           MVI   A,0FFH
1634 32 35 1D    0335           STA   LOCSTK+160  MARK END OF CMD STACK
1637 3A 03 16    0340           LDA   STATPRT  "ADDRESS" OF KEYBOARD STATUS PORT
163A 32 49 19    0345           STA   GETSTAT+1
163D 3A 04 16    0350           LDA   KBDMSK   KEYBOARD READY BIT
1640 32 4B 19    0355           STA   GETSTAT+3
1643 C9          0360           RET
1644             0365 *
1644             0370 *THESE ROUTINES IMPLEMENT COMMAND STACK AND DISK SPOOL
1644             0375 *
1644             0380 * IOFLAG BITS HAVE FOLLOWING MEANINGS WHEN SET
1644             0385 * BIT 0 - ALL OUTPUT TO PRINTER AND CRT
1644             0390 * BIT 1 - PRINT SPOOL IS ACTIVE
1644             0395 * BIT 2 - ALL OUTPUT TO WRITE SPOOL AND CRT
1644             0400 * BIT 3 - ALL INPUT FROM READ SPOOL (KEYBOARD DISABELED
1644             0405 *           EXECEPT FOR ^C)
1644             0410 * BIT 4 - IGNORE DEVICE NUMBER IN A-REG (USED FOR
1644             0415 *           PROGRAMS WHICH DO NOT FOLLOW DOS CONVENTIONS
1644             0420 *           FOR USE OF A-REG TO SIGNAL DEVICE NUMBER)
1644             0425 * BIT 7 - SPOOL ERROR, READ/WRITE SPOOL DISABLED
1644             0430 *
1644             0435 *INPUT ROUTINE, READ A CHARACTER FROM KEYBOARD
1644             0440 *OR FROM COMMAND STACK OR FROM DISK SPOOL
1644             0445 *
1644 E5          0450 SPLCIN:   PUSH  H         CHANGE STACK POINTER
1645 21 00 00    0455           LXI   H,0       SO SUFFICIENT SPACE WILL BE
1648 39          0460           DAD   SP        AVAILABLE FOR DISK USEAGE
1649 31 FF 1C    0465           LXI   SP,IOSTK NEW STACK AREA
164C E5          0470           PUSH  H         SAVE OLD STACK POINTER
164D D5          0475           PUSH  D         AND ALL REGS
164E C5          0480           PUSH  B
164F 32 55 1C    0485           STA   IODEV     SAVE OUTPUT DEVICE CODE
1652 CD 5F 16    0490 LOOP:     CALL  FETCH     INTERNAL LOOP FOR COMMON RET
1655 FA 52 16    0495 TESTS:    JM    LOOP      BIT 7 MEANS CONT INTERNAL LOOP
1658 C1          0500           POP   B         RESTORE REGISTERS
1659 D1          0505 FIN:      POP   D
165A E1          0510           POP   H         GET OLD STACK POINTER
165B F9          0515           SPHL            RESTORE STACK POINTER
165C E1          0520           POP   H
165D B7          0525           ORA   A
165E C9          0530           RET             A-REG CONTAINS CHAR FOR CLNG PROG
165F             0535 *
165F             0540 * ROUTINE TO GET A CHAR(S) FROM READ SPOOL OR KEYBOARD
165F             0545 * OR COMMAND STACK AND CHECK FOR CONTROL CHARACTERS TO
165F             0550 * START READ/WRITE/PRINT SPOOLS OR INTELLIGINT TERMINAL
165F             0555 *
165F 11 50 1C    0560 FETCH:    LXI   D,RDSTK   SET UP REGS FOR COMMAND STACK
1662 2A 50 1C    0565           LHLD  RDSTK
1665 7C          0570           MOV   A,H       SEE IF COMMAND STACK
1666 B5          0575           ORA   L         IS ACTIVE
1667 C2 2B 17    0580           JNZ   GOSTK     BRIF ACTIVE
166A 21 54 1C    0585           LXI   H,IOFLAG POINT TO I/O CONTROL
166D 11 6F 1C    0590           LXI   D,RDFILE D,E TO READ SPOOL NAME
1670 7E          0595           MOV   A,M       CHECK READ SPOOL FLAG
1671 E6 88       0600           ANI   88H       BIT 7 IS ERROR, BIT 3 IS READ
1673 FA 79 16    0605           JM    $+3       SKIP SPOOL IF ERROR BIT SET
1676 C2 86 17    0610           JNZ   GETSPL    BRIF IF SPOOL IS 'ON'
1679 7E          0615           MOV   A,M       CHECK IF INPUT DEVICE
167A E6 10       0620           ANI   10H       NUMBER SHOULD BE USED
167C C2 8C 16    0625           JNZ   KEYBD     IGNORE DEV NUM IF SET
167F 7E          0630           MOV   A,M       CHECK SPOOL ERROR BIT
1680 B7          0635           ORA   A
1681 FA 8C 16    0640           JM    KEYBD     SKIP SPOOL IF ERROR BIT SET
1684 3A 55 1C    0645           LDA   IODEV     CHECK INPUT DEVICE #
1687 FE 08       0650           CPI   8         DEVICE 8 FOR READ SPOOL
1689 CA 86 17    0655           JZ    GETSPL
168C 11 82 1C    0660 KEYBD:    LXI   D,PRFILE GET CHAR FROM KBD, CHECK FOR ESC
168F CD F5 17    0665           CALL  GETCP     CHECK IF PRINT SPOOL IS ACTIVE
1692 11 50 1C    0670           LXI   D,RDSTK   INIT D,E IF STACK COMMAND
1695 2A 52 1C    0675           LHLD  STACK     INIT H,L IF STACK COMMAND
1698 FE 7C       0680           CPI   '|'       LOOK FOR START OF STACK
```

```
169A CA 44 17    0685           JZ    LDSTK
169D B7          0690           ORA   A         LOOK FOR ESCAPE SEQUENCE
169E F0          0695           RP              RETURN TO CALLING PROG IF NONE
169F E6 7F       0700           ANI   7FH       STRIP OFF FLAG
16A1 FE 7C       0705           CPI   '|'       LOOK FOR CHAR '|'
16A3 C8          0710           RZ              SEND TO CALLING PROG
16A4 FE 1B       0715           CPI   ESC       LOOK FOR ESC CHAR
16A6 C8          0720           RZ
16A7 FE 13       0725           CPI   CTRLS     LOOK FOR EXECUTE STACK
16A9 CA 2B 17    0730           JZ    GOSTK     EXECUTE THE STACK
16AC 11 5C 1C    0735           LXI   D,SPLFILE  PREPARE TO CHANGE FILE NAMES
16AF FE 52       0740           CPI   'R'       READ FILE NAME
16B1 CA 6C 18    0745           JZ    RDNAME
16B4 FE 4F       0750           CPI   'O'       PRINT (OUTPUT) FILE NAME
16B6 CA 7B 18    0755           JZ    PRNAME
16B9 FE 57       0760           CPI   'W'       WRITE FILE NAME
16BB CA 74 18    0765           JZ    WRNAME
16BE 21 54 1C    0770           LXI   H,IOFLAG
16C1 FE 03       0775           CPI   CTRLC     LOOK FOR CLOSE WRITE SPOOL
16C3 CA 52 18    0780           JZ    CLSPOOL
16C6 FE 30       0785           CPI   '0'       LOOK FOR IOFLAG CONTROL
16C8 DA 79 1B    0790           JC    ESCERR
16CB FE 41       0795           CPI   'A'
16CD F2 79 1B    0800           JP    ESCERR
16D0 D6 30       0805           SUI   '0'       CONVERT CHAR TO BINARY
16D2 CA DB 16    0810           JZ    SKIP      INIT IOFLAG, CLEAR ERROR BIT
16D5 E6 1F       0815           ANI   1FH       ONLY BITS 0-4 IMPLEMENTED
16D7 EA 79 1B    0820           JPE   ESCERR    ONLY ONE BIT AT A TIME
16DA B6          0825           ORA   M         PRESERVE OTHER BIT
16DB 77          0830 SKIP:     MOV   M,A
16DC F6 80       0835           ORI   80H       RESTORE FLAG
16DE C9          0840           RET
16DF             0845 *
16DF             0850 *THIS ROUTINE RECEIVES A CHAR (B-REG) AND
16DF             0855 *DEVICE CODE (A-REG) AND OUTPUTS CHAR
16DF             0865 *
16DF FE 08       0870 SPLCOUT:  CPI   8         DON'T ECHO READ SPOOL
16E1 C8          0875           RZ              CHARACTERS
16E2 E5          0880           PUSH  H         CHANGE STACK POINTER
16E3 21 00 00    0885           LXI   H,0       SO SUFFICIENT SPACE WILL BE
16E6 39          0890           DAD   SP        AVAILABLE FOR DISK USEAGE
16E7 31 FF 1C    0895           LXI   SP,IOSTK NEW STACK AREA
16EA E5          0900           PUSH  H         SAVE OLD STACK POINTER
16EB D5          0905           PUSH  D         SAVE ALL REGS
16EC C5          0910           PUSH  B
16ED 4F          0915           MOV   C,A       SAVE OUTPUT DEVICE #
16EE 21 54 1C    0920           LXI   H,IOFLAG
16F1 32 55 1C    0925           STA   IODEV     SAVE SELECTED OUTPUT DEVICE
16F4 CD 16 17    0935           CALL  OUTSPL    CHECK IF OUTPUT SPOOL
16F7 C1          0940           POP   B         IS ACTIVE; RESTORE CHAR
16F8 7E          0950           MOV   A,M
16F9 E6 10       0955           ANI   10H       LOOK FOR CRT ONLY FLAG
16FB C2 0F 17    0960           JNZ   DOCRT     IGNORE DEVICE NUMBER IF SET
16FE 7E          0965           MOV   A,M
16FF E6 01       0970           ANI   1         LOOK FOR PRINTER FLAG
1701 C2 0C 17    0975           JNZ   DOPRT
1704 23          0980           INX   H         POINT TO IODEV
1705 7E          0985           MOV   A,M       RESTORE DEVICE CODE
1706 CD 05 16    0990           CALL  USRCOUT
1709 C3 59 16    0995           JMP   FIN
170C CD 58 19    1000 DOPRT:    CALL  PUTCR
170F 78          1005 DOCRT:    MOV   A,B
1710 CD 4F 19    1006           CALL  DISPLAY
1713 C3 59 16    1010           JMP   FIN
1716             1015 *
1716             1020 * CHECK IF WRITE SPOOL IS ACTIVE OR IF DEVICE 4 IS REQ
1716             1025 * WRITE CHARACTER TO DISK SPOOL
1716             1030 *
1716 7E          1035 OUTSPL:   MOV   A,M       CHECK WRITE SPOOL FLAG
1717 E6 84       1040           ANI   84H       BIT 7 IS ERROR, BIT 2 IS WRITE
1719 F8          1045           RM              SKIP WRITE SPOOL IF ERROR
171A 11 5C 1C    1050           LXI   D,WRFILE SET UP POINTER
171D C2 B7 17    1055           JNZ   PUTSPL    BRIF WRITE SPOOL IS 'ON'
1720 7E          1060           MOV   A,M       CHECK DEVICE NUMBER
1721 E6 10       1065           ANI   10H       IGNORE BIT
1723 C0          1070           RNZ
1724 79          1075           MOV   A,C
1725 FE 04       1080           CPI   4         ELSE CHECK DEVICE #
1727 CA B7 17    1085           JZ    PUTSPL    BRIF DEVICE 4; WRITE SPOOL
172A C9          1090           RET
172B             1095 *
172B             1100 *THESE ROUTINES HANDLE THE COMMAND STACK. THE ROUTINES
172B             1105 *USE 2 BYTES OF STORAGE IN THE DOS-I/O PROGRAM TO
172B             1110 *SAVE A POINTER TO THE COMMAND STACK.
172B             1115 *
172B             1120 * ENTRY TO EXECUTE STACK MODE
172B             1125 * H,L POINTS TO CHARACTER ON STACK
172B             1130 *
172B 7E          1135 GOSTK:    MOV   A,M
172C 23          1140           INX   H         POINT TO NEXT CHAR
172D FE 0D       1145           CPI   CR        LOOK FOR END OF STACK
172F CA 3C 17    1150           JZ    KILL      BRIF CR, KILL STACK MODE
1732 FE 7C       1155           CPI   '|'       LOOK FOR END OF LINE
1734 C2 3F 17    1160           JNZ   SAVPTR    BRIF NOT EOL, RET CHAR
1737 3E 0D       1165           MVI   A,CR      RETURN A CR FOR EOL
1739 C3 3F 17    1170           JMP   SAVPTR    AND SAVE POINTER
173C 21 00 00    1175 KILL:     LXI   H,0       KILL STACK MODE
173F E6 7F       1180 SAVPTR:   ANI   7FH       CLEAR ESCAPE FLAG
1741 C3 26 19    1185           JMP   UPDATE    H,L -> (D,E) AND RETURN
1744             1190 *
1744             1195 * ENTRY TO LOAD STACK
1744             1200 * H,L POINTS TO START OF STACK
1744             1205 *
1744 CD 26 19    1210 LDSTK:    CALL  UPDATE    SAVE INITIAL VALUE OF POINTER
1747 CD 4F 19    1215           CALL  DISPLAY   ECHO CHAR
174A CD 36 19    1220 LOAD:     CALL  GETCHR    GET CHAR AND FLAG ESC
174D CD 4F 19    1225           CALL  DISPLAY   ECHO CHAR
1750 77          1230           MOV   M,A       SAVE CHARACTER
1751 FE 0D       1235           CPI   CR        LOOK FOR CR
1753 C8          1240           RZ              RETURN IF CR, END OF STACK
1754 FE 03       1245           CPI   CTRLC     CHECK FOR KILL
1756 CA 80 17    1250           JZ    KILLSTK   DELETE STACK AND RETURN ^C
1759 FE 08       1255           CPI   BS        CHECK CHAR DELETE
175B CA 73 17    1260           JZ    DELETE
175E FE 20       1265           CPI   ' '       LOOK FOR CONTROL CHARS
1760 DA 4A 17    1270           JC    LOAD      NOT ALLOWED ON STACK
1763 23          1275           INX   H         BUMP POINTER TO STACK
1764 C5          1280           PUSH  B
1765 46          1285           MOV   B,M       CHECK FOR END OF STACK FLAG
1766 04          1290           INR   B
1767 C1          1295           POP   B
1768 C2 4A 17    1300           JNZ   LOAD      CONTINUE IF NOT END
176B CD 4D 19    1305           CALL  DSPBELL   INDICATE END OF STACK
176E 3E 08       1310           MVI   A,BS
1770 CD 4F 19    1315           CALL  DISPLAY   ECHO BS
1773 3E 7F       1320 DELETE:   MVI   A,RUBOUT DELETE LAST CHAR
1775 CD 4F 19    1325           CALL  DISPLAY
1778 1A          1330           LDAX  D         GET INITIAL VALUE OF POINTER
1779 BD          1335           CMP   L         SEE IF CURRENTLY AT FIRST POS
177A 2B          1340           DCX   H         ADJUST STACK POINTER
177B C2 4A 17    1345           JNZ   LOAD      CONTINUE IF CHARS REMAIN
177E F6 80       1350           ORI   80H       ELSE SET FLAG AND RETURN
1780 21 00 00    1355 KILLSTK:  LXI   H,0       KILL STACK MODE AND RETURN
1783 C3 26 19    1357           JMP   UPDATE
1786             1360 *
1786             1365 *THESE ROUTINES ARE USED BY THE CIN AND COUT I/O PROGS
1786             1370 *TO READ AND WRITE THE DISK SPOOL. ROUTINES USE 57 BYTES
```

```
1786             *OF STORAGE FOR THE FOLLOWING ITEMS:
1786             * 1) XXFILE (12) - CONTAINS THE NAME OF THE SPOOL FILE
1786             * 2) XXUNIT (1)  - CONTAINS THE NUMBER OF THE DRIVE FOR
1786             *                   THE ABOVE SPOOL FILE
1786             * 3) XXPTR  (1)  - POINTER TO RAM BUFFER (LOW BYTE)
1786             * 4) XXBUF  (1)  - POINTER TO RAM BUFFER (HIGH BYTE)
1786             * 5) XXADR  (2)  - DISK ADDRESS FOR NEXT READ/WRITE
1786             * 6) XXLEN  (2)  - DISK FILE LENGTH
1786             *WHERE XX IF WR IF WRITE SPOOL, RD IF READ SPOOL, AND
1786             *PR IF PRINT SPOOL.
1786             *THE SPOOL DRIVERS ENTER THESE ROUTINES WITH ALL
1786             *NECESSARY INFORMATION IN 8080 REGISTERS.
1786             *
1786             * DRIVER ROUTINE FOR SPOOL READ
1786             * H,L -> IOFLAG; D,E -> RDFILE
1786             *
1786 E5          GETSPL:  PUSH  H        SAVE ADDR OF IOFLAG
1787 21 0C 00             LXI   H,12     D,E+12->H,L
178A 19                   DAD   D
178B EB                   XCHG           H,L -> RDFILE; D,E -> RDUNIT
178C 1A                   LDAX  D        CHECK IF READ SPOOL IS OPEN
178D B7                   ORA   A
178E CC 44 18             CZ    SPLSRCH  OPEN READ SPOOL IF NECESSARY
1791 DA 43 1B             JC    RDERR    BRIF OPEN FAILS
1794 CD 2D 19             CALL  LOADHL   H,L -> RAM BUFFER
1797 CD 7F 19             CALL  RDDISK   READ ONE CHAR FROM SPOOL
179A CA 39 1C             JZ    STOPR    STOP SPOOL IF END-OF-FILE
179D DA 3D 1B             JC    LNERR    LENGTH ERROR
17A0 E1                   POP   H        H,L -> IOFLAG
17A1 FE 0D                CPI   CR       LOOK FOR END-OF-LINE
17A3 CA A9 17             JZ    CKEYBD   BRIF CR
17A6 E6 7F                ANI   7FH      TURN OFF BIT 7
17A8 C9                   RET
17A9             *
17A9             * CHECK KEYBOARD AT END-OF-LINE WHEN READ SPOOL
17A9             * IS ACTIVE.  HALT READ SPOOL IF ANY CHARACTER IS
17A9             * TYPED.
17A9             *
17A9 CD 48 19    CKEYBD:  CALL  GETSTAT
17AC 3E 0D                MVI   A,CR     RESTORE CR
17AE C8                   RZ             RETURN IF NO CHAR TYPED
17AF 7E                   MOV   A,M      GET I/O CONTROL
17B0 F6 80                ORI   80H      SET ERROR BIT
17B2 77                   MOV   M,A
17B3 AF                   XRA   A        RESET FLAGS
17B4 3E 0D                MVI   A,CR     RESTORE CR
17B6 C9                   RET
17B7             *
17B7             * DRIVER ROUTINE FOR SPOOL WRITE
17B7             * H,L -> IOFLAG; D,E -> WRFILE
17B7             * B-REG CONTAINS CHAR TO OUTPUT
17B7             *
17B7 3E 0A       PUTSPL:  MVI   A,LF     SKIP LF CHARS
17B9 B8                   CMP   B
17BA C8                   RZ
17BB 3E 01                MVI   A,1      LOOK FOR END-OF-FILE
17BD B8                   CMP   B        CLOSE WRITE SPOOL WHEN SOH
17BE CA 52 18             JZ    CLSPOOL  (ASCII 1) IS OUTPUT
17C1 E5                   PUSH  H        SAVE ADDR OF IOFLAG
17C2 21 0C 00             LXI   H,12     D,E+12->H,L
17C5 19                   DAD   D        H,L -> WRUNIT
17C6 7E                   MOV   A,M      SEE IF WRITE SPOOL IS OPEN
17C7 EB                   XCHG           H,L -> WRFILE; D,E -> WRUNIT
17C8 B7                   ORA   A
17C9 CC 44 18             CZ    SPLSRCH  LOOK FOR SPOOL FILE IF NOT OPEN
17CC C5                   PUSH  B        SAVE B,C
17CD 01 00 00             LXI   B,0      FLAG FOR CREATE ROUTINE
17D0 3E 07                MVI   A,7      SET FILE TYPE
17D2 DC F3 19             CC    CREATE   CREATE FILE IF NOT FOUND
17D5 C1                   POP   B        RESTORE B,C
17D6 D2 E4 17             JNC   OK1      BRIF NO ERROR
17D9 CA 4A 1B             JZ    FLERR    DISK FULL
17DC FE 01                CPI   1
17DE CA 51 1B             JZ    NMERR    FILE NAME ERROR
17E1 C3 37 1B             JMP   WRERR    DISK WRITE PROTECT ERROR
17E4 CD 2D 19    OK1:     CALL  LOADHL   H,L -> RAM BUFFER
17E7 CD 5D 19             CALL  WRDISK   WRITE CHAR TO SPOOL
17EA D2 F3 17             JNC   OK2      BRIF NO ERROR
17ED CA 3D 1B             JZ    LNERR    LENGTH ERROR
17F0 C3 37 1B             JMP   WRERR    WRITE PROTECT
17F3 E1          OK2:     POP   H        CLEAR ADDR OF IOFLAG
17F4 C9                   RET
17F5             *
17F5             * DRIVER ROUTINE FOR PRINT SPOOL
17F5             * H,L -> IOFLAG; D,E -> PRFILE
17F5             *
17F5 7E          GETCP:   MOV   A,M      SEE IF PRINT SPOOL IS ACTIVE
17F6 E6 02                ANI   2        BIT 1 IS FLAG
17F8 CA 36 19             JZ    GETCHAR  BRIF NOT ACT, GET CHAR FROM KBD
17FB CD 07 18             CALL  PRACT    PRINT SPOOL IS ACTIVE
17FE F2 36 19             JP    GETCHAR  BRIF NO ERROR, GET CHAR FROM KBD
1801 E6 1D                ANI   1DH      ERROR OR END OF PRINT SPOOL
1803 77                   MOV   M,A      KILL BIT 1 & ERROR BIT, SAVE REST
1804 C3 36 19             JMP   GETCHAR  GO TO KEYBOARD
1807 E5          PRACT:   PUSH  H        SAVE ADDR OF IOFLAG
1808 21 0C 00             LXI   H,12     D,E+12->H,L
180B 19                   DAD   D        H,L -> PRUNIT
180C 7E                   MOV   A,M      SEE IF PRINT FILE IS OPEN
180D EB                   XCHG           H,L -> PRFILE; D,E -> PRUNIT
180E B7                   ORA   A
180F CC 44 18             CZ    SPLSRCH  LOOK FOR SPOOL FILE IF NOT OPEN
1812 DA 43 1B             JC    RDERR    BRIF IF NOT FOUND
1815 CD 2D 19             CALL  LOADHL   H,L -> RAM BUFFER
1818 7D          PRLOOP:  MOV   A,L      LOOK FOR END OF BUFFER
1819 FE DA                CPI   -38      START DRIVE MOTORS
181B CC 16 1B             CZ    STRTM    WHEN 38 CHARS LEFT
181E FE F8                CPI   -8       SELECT DRIVE, LOAD HEAD
1820 1A                   LDAX  D        WHEN 8 CHARS LEFT
1821 4F                   MOV   C,A
1822 06 EB                MVI   B,CC
1824 CC 2C 1B             CZ    SELECT
1827 CD 7F 19             CALL  RDDISK   READ ONE CHAR FROM SPOOL
182A CA 39 1C             JZ    STOPR    BRIF END-OF-FILE
182D DA 3D 1B             JC    LNERR    LENGTH ERROR IF 'C' SET
1830 47                   MOV   B,A
1831 CD 58 19             CALL  PUTCR    OUTPUT CHAR TO PRINTER
1834 3E 0D                MVI   A,CR     LOOK FOR CR
1836 B8                   CMP   B        ADD A LINE FEED
1837 06 0A                MVI   B,LF
1839 CC 58 19             CZ    PUTCR
183C CD 48 19             CALL  GETSTAT  CHECK KEYBOARD STATUS
183F CA 18 18             JZ    PRLOOP   BRIF NO CHAR READY
1842 E1                   POP   H        ELSE CLEAR IOFLAG FROM STACK
1843 C9                   RET            AND RETURN
1844             *
1844             * SEARCH DIRECTORY FOR SPOOL FILE
1844             * H,L -> XXFILE (FILE NAME)
1844             * D,E -> XXUNIT (DISK UNIT #)
1844             * RETURN WITH 'C' SET IF FILENAME NOT FOUND
1844             *
1844 CD D3 19    SPLSRCH: CALL  SEARCH
1847 CA 50 1B             JZ    NPERR    BAD FILENAME
184A D8                   RC
184B FE 07                CPI   7        CHECK FOR PROPER FILE TYPE
184D C8                   RZ             RETURN, NO ERROR
184E F1                   POP   PSW      CLEAR RETURN ADDRESS
184F C3 57 1B             JMP   TYERR    SIGNAL TYPE ERROR
```

```
1852             *
1852             * CLOSE WRITE SPOOL
1852             * H,L -> IOFLAG; D,E -> WRFILE
1852             *
1852 E5          CLSPOOL: PUSH  H        SAVE ADDR OF IOFLAG
1853 7E                   MOV   A,M
1854 E6 8B                ANI   8BH      KILL BIT 2, SAVE ERROR FLAG
1856 77                   MOV   M,A      UPDATE IOFLAG
1857 21 0C 00             LXI   H,12     D,E+12->H,L
185A 19                   DAD   D
185B EB                   XCHG           H,L -> WRFILE; D,E -> WRUNIT
185C CD 8A 1A             CALL  CLSIT    CLOSE SPOOL IF OPEN
185F DA 5D 1B             JC    CLSERR
1862 21 DC 1B             LXI   H,CLOSED PRINT 'CLOSED' MESSAGE
1865 CD 2E 1C             CALL  PMSG
1868 E1                   POP   H        RESTORE STACK
1869 F6 80                ORI   80H      SET 'M' FLAG
186B C9                   RET
186C             *
186C             * CHANGE NAME OF READ SPOOL FILE - ESC 'R'
186C             * CHANGE NAME OF WRITE SPOOL FILE- ESC 'W'
186C             * CHANGE NAME OF PRINT FILE      - ESC 'O'
186C             * D,E -> SPLFILE
186C             *
186C 21 DC 18    RDNAME:  LXI   H,READN
186F 3E 13                MVI   A,19     OFFSET FOR RDFILE
1871 C3 80 18             JMP   GNAME
1874 21 E8 18    WRNAME:  LXI   H,WRITEN
1877 AF                   XRA   A        OFFSET FOR WRFILE
1878 C3 80 18             JMP   GNAME
187B 21 F5 18    PRNAME:  LXI   H,PRINTN
187E 3E 26                MVI   A,38     OFFSET FOR PRFILE
1880 D5          GNAME:   PUSH  D        D,E -> SPLFILE
1881 83                   ADD   E        ADD OFFSET TO XXFILE
1882 5F                   MOV   E,A      D,E -> XXFILE
1883 D2 87 18             JNC   $+1
1886 14                   INR   D        BUMP D-REG IF NECESSARY
1887 CD 2E 1C             CALL  PMSG     PRINT SPOOL TYPE MESSAGE
188A E1                   POP   H        H,L -> SPLFILE
188B 2B                   DCX   H        SAVE POINTER TO XXFILE
188C 72                   MOV   M,D      FOR LOAD ROUTINE
188D 2B                   DCX   H
188E 73                   MOV   M,E
188F E5                   PUSH  H
1890 21 01 19             LXI   H,CURR
1893 CD 2E 1C             CALL  PMSG
1896 62                   MOV   H,D      PRINT CURRENT SPOOL FILE NAME
1897 6B                   MOV   L,E
1898 7E          NAME:    MOV   A,M
1899 CD 4F 19             CALL  DISPLAY
189C 23                   INX   H
189D FE 0D                CPI   CR       STOP ON CR
189F CA AD 18             JZ    OKNAME   BRIF PROPER TERM
18A2 FE 20                CPI   ' '
18A4 D2 98 18             JNC   NAME     BRIF IF VALID CHAR
18A7 21 CC 1B             LXI   H,BADNM  SET MSG FOR BAD FILE NAME
18AA CD 2E 1C             CALL  PMSG     PRINT IT
18AD 21 13 19    OKNAME:  LXI   H,NEW
18B0 CD 2E 1C             CALL  PMSG     PRINT NEW PROMPT
18B3 CD 36 19             CALL  GETCHR   GET FIRST CHAR
18B6 FE 0D                CPI   CR       TO SEE IF NAME SHOULD BE CHANGED
18B8 E1                   POP   H
18B9 CA D3 18             JZ    RETM     BRIF NO CHANGE DESIRED
18BC E5                   PUSH  H
18BD 21 0C 00             LXI   H,12     D,E+12->H,L
18C0 19                   DAD   D        H,L -> XXUNIT
18C1 36 00                MVI   M,0      INDICATE FILE IS 'CLOSED'
18C3 EB                   XCHG           H,L -> XXFILE
18C4 D1                   POP   D        D,E -> SPLFILE-2
18C5 CD 4D 17             CALL  LOAD+3   USE LOAD STACK ROUTINE TO
18C8 21 CC 1B             LXI   H,BADNM  ENTER NEW NAME
18CB FC 2E 1C             CM    PMSG
18CE FE 03                CPI   CTRLC
18D0 CC 2E 1C             CZ    PMSG
18D3 21 ED 1B    RETM:    LXI   H,CRLF
18D6 CD 2E 1C             CALL  PMSG     NEW LINE
18D9 F6 80                ORI   80H      SET 'M' FLAG
18DB C9                   RET
18DC 52 45 41    READN:   ASC   'READ SPOOL '
     44 20 53 50 4F 4F 4C 20
18E7 00                   DB    0
18E8 57 52 49    WRITEN:  ASC   'WRITE SPOOL '
     54 45 20 53 50 4F 4F 4C 20
18F4 00                   DB    0
18F5 50 52 49    PRINTN:  ASC   'PRINT FILE '
     4E 54 20 46 49 4C 45 20
1900 00                   DB    0
1901 43 55 52    CURR:    ASC   'CURRENT NAME IS: '
     52 45 4E 54 20 4E 41 4D 45 20
     49 53 3A 20
1912 00                   DB    0
1913 0D          NEW:     DB    CR
1914 0A                   DB    LF
1915 45 4E 54             ASC   'ENTER NEW NAME: '
     45 52 20 4E 45 57 20 4E 41 4D
     45 3A 20
1925 00                   DB    0
1926             *
1926             * MISCELLANEOUS SUBROUTINES
1926             *
1926             * SIMULATE A SHLD AT ADDRESS IN D,E
1926             *
1926 EB          UPDATE:  XCHG
1927 73                   MOV   M,E      SAVE LOW BYTE
1928 23                   INX   H
1929 72                   MOV   M,D      SAVE HIGH BYTE
192A 2B                   DCX   H
192B EB                   XCHG
192C C9                   RET
192D             *
192D             * SIMULATE A LHLD FROM ADDRESS IN D,E+1
192D             *
192D EB          LOADHL:  XCHG
192E E5                   PUSH  H
192F 23                   INX   H
1930 5E                   MOV   E,M
1931 23                   INX   H
1932 56                   MOV   D,M
1933 E1                   POP   H
1934 EB                   XCHG
1935 C9                   RET
1936             *-
1936             *
1936             * GET A CHAR FROM KBD, IF ESCAPE THEN GET ANOTHER CHAR
1936             * AND FLAG BIT 7 TO INDICATE "ESCAPE SEQUENCE"
1936             *
1936 3A 55 1C    GETCHR:  LDA   IODEV    GET A CHAR FROM
1939 CD 08 16             CALL  USRCIN   SELECTED DEVICE
193C FE 1B                CPI   ESC      LOOK FOR "SPOOL CONTROL"
193E C0                   RNZ
193F 3A 55 1C             LDA   IODEV    GET ANOTHER CHARACTER
1942 CD 08 16             CALL  USRCIN
1945 F6 80                ORI   80H      SET CONTROL FLAG
1947 C9                   RET
1948             *
1948             *GET STATUS OF SELECTED INPUT DEVICE
```

```
1948            2685 *THIS ROUTINE MUST BE MODIFIED TO PARTICULER
1948            2690 *I/O CONFIGURATION. RETURN WITH Z-FLAG SET IF
1948            2695 *NO CHARACTER IS READY ON KEYBOARD
1948            2700 *** NOTE: THE 0'S ARE CHANGED BY THE DOSPOOL ROUTINE
1948            2705 *
1948 DB 00      2710 GETSTAT: IN    0        "ADDRESS" OF KEYBOARD STATUS PORT
194A E6 00      2715          ANI   0        KEYBOARD READY BIT
194C C9         2720          RET
194D            2725 *
194D            2730 *RING BELL ON TERMINAL
194D            2735 *
194D 3E 07      2740 DSPBELL: MVI   A,BELL
194F            2745 *
194F            2750 *OUTPUT CHARACTER (A-REG) TO CRT (DEVICE 0)
194F            2755 *
194F C5         2760 DISPLAY: PUSH  B
1950 47         2765          MOV   B,A      GET CHARACTER IN B-REG
1951 3E 00      2770          MVI   A,0      SELECT DEVICE 0
1953 CD 05 16   2775          CALL  USRCOUT
1956 C1         2780          POP   B        RESTORE B
1957 C9         2785          RET
1958            2790 *
1958            2795 *OUTPUT CHARACTER (B-REG) TO PRINTER (DEVICE 1)
1958            2800 *
1958 3E 01      2805 PUTCR:   MVI   A,1
195A C3 05 16   2810          JMP   USRCOUT
195D            2815 *
195D            2820 *THESE ROUTINES DO DISK I/O FOR THE DISK AND SPOOL
195D            2825 *DRIVERS.  ALL POINTERS ARE PASSED IN THE REGISTERS AS
195D            2830 *INDICATED AT THE BEGINNING OF EACH ROUTINE.
195D            2835 *
195D            2840 * ENTRY TO WRITE CHAR IN B-REG TO DISK.
195D            2845 * H,L -> RAM BUFFER
195D            2850 * D,E -> WRUNIT
195D            2855 * RETURN WITH 'C' SET IF ERROR
195D            2860 * IF ERROR: 'Z' SET FOR LENGTH ERROR,
195D            2865 *                RESET FOR DISK WRITE PROTECT
195D            2870 *
195D 70         2875 WRDISK:  MOV   M,B      SAVE CHAR IN BUFFER
195E EB         2880          XCHG
195F 3E 80      2885          MVI   A,80H    CHECK IF HALF-WAY THROUGH
1961 BB         2890          CMP   E        BUFFER
1962 CC 22 1B   2895          CZ    CKMOTOR  BRIF HALF-WAY
1965 23         2900          INX   H        POINT TO WRBUF
1966 AF         2905          XRA   A        CLEAR ALL FLAGS
1967 1C         2910          INR   E        UPDATE POINTER
1968 73         2915          MOV   M,E      LOW BYTE
1969 2B         2920          DCX   H        RESTORE H,L AND
196A EB         2925          XCHG           D,E TO VALUE AT ENTRY
196B C0         2930          RNZ            RETURN IF BUFFER IS NOT FULL
196C E5         2935          PUSH  H        ELSE WRITE BUFFER TO DISK
196D D5         2940          PUSH  D        SAVE REGS
196E EB         2945          XCHG
196F 7E         2950          MOV   A,M      GET DISK DRIVE #
1970 E6 03      2955          ANI   3        REMOVE POSSIBLE 'M' FLAG
1972 4F         2960          MOV   C,A      SET UP REGISTERS
1973 23         2965          INX   H        SKIP WRBUF ADDRESS
1974 23         2970          INX   H        H,L -> WRBUF
1975 06 00      2975          MVI   B,0      SET DISK WRITE COMMAND
1977 3E 01      2980          MVI   A,1      SET LENGTH TO 1 BLOCK
1979 CD A8 19   2985          CALL  DODSK    WRITE BLOCK TO DISK
197C D1         2990          POP   D        D,E -> WRUNIT
197D E1         2995          POP   H        H,L -> RAM BUFFER
197E C9         3000          RET            'C' FLAG SET IF ERROR
197F            3005 *
197F            3010 * ENTRY TO READ A CHAR FROM DISK FILE.
197F            3015 * SAME ENTRY FOR BOTH DISK AND SPOOL DRIVERS.
197F            3020 * H,L -> RAM BUFFER
197F            3025 * D,E -> RDUNIT
197F            3030 *
197F            3035 * RETURN CHARACTER IN A-REG.
197F            3040 * 'Z' FLAG SET IF EOF (1); 'C' FLAG SET IF LENGTH ERROR.
197F            3045 *
197F AF         3050 RDDISK:  XRA   A        TEST LOW BYTE OF BUFFER
1980 B5         3055          ORA   L        ADDRESS FOR ZERO
1981 C2 96 19   3060          JNZ   GETIT    ZERO MEANS ANOTHER BLOCK
1984 E5         3065          PUSH  H        SHOULD BE READ FROM THE
1985 D5         3070          PUSH  D        DISK SPOOL
1986 EB         3075          XCHG           H,L -> RDUNIT
1987 7E         3080          MOV   A,M      GET DISK DRIVE #
1988 E6 03      3085          ANI   3        REMOVE POSSIBLE 'M' FLAG
198A 4F         3090          MOV   C,A      SET UP REGISTERS
198B 06 01      3095          MVI   B,1      SET READ COMMAND
198D 78         3100          MOV   A,B      SET LENGTH TO 1 BLOCK
198E 23         3105          INX   H        POINT TO DISK ADDRESS
198F 23         3110          INX   H
1990 CD A8 19   3115          CALL  DODSK    GET ANOTHER BLOCK
1993 D1         3120          POP   D        D,E -> RDUNIT
1994 E1         3125          POP   H        H,L -> RAM BUFFER
1995 D8         3130          RC             'C' SET FOR LENGTH ERROR
1996 3E 80      3135 GETIT:   MVI   A,80H    CHECK IF HALF-WAY
1998 BD         3140          CMP   L        THROUGH BUFFER
1999 CC 22 1B   3145          CZ    CKMOTOR  BRIF HALF-WAY
199C 7E         3150          MOV   A,M      GET CHAR FROM BUFFER
199D 2C         3155          INR   L        SET NEXT BUFFER ADDRESS
199E 13         3160          INX   D        POINT TO RDPTR
199F CD 26 19   3165          CALL  UPDATE   SAVE NEW BUFFER ADDRESS
19A2 1B         3170          DCX   D        POINT TO RDUNIT
19A3 FE 01      3175          CPI   1        LOOK FOR END-OF-SPOOL
19A5 D0         3180          RNC            'Z' FLAG SET IF EOF
19A6 3F         3185          CMC            RESET 'C' FLAG IF NULL CHAR
19A7 C9         3190          RET            READ FROM DISK
19A8            3195 *
19A8            3200 * DISK COMMAND SUBROUTINE TO READ OR WRITE
19A8            3205 * ONE BLOCK OF THE DISK FILE
19A8            3210 * H,L -> XXBUF
19A8            3215 * D,E -> START OF RAM BUFFER
19A8            3220 * B CONTAINS DISK COMMAND CODE
19A8            3225 * C CONTAINS DISK UNIT NUMBER
19A8            3230 * A CONTAINS # OF BLOCKS TO READ/WRITE
19A8            3235 * RETURN WITH 'C' SET IF ERROR
19A8            3240 * IF ERROR: 'Z' SET FOR LENGTH ERROR,
19A8            3245 *                RESET FOR DISK WRITE PROTECT
19A8            3250 *
19A8 F5         3255 DODSK:   PUSH  PSW      SAVE LENGTH
19A9 D5         3260          PUSH  D        SAVE BUFFER ADDR
19AA 23         3265          INX   H        POINT TO XXADR
19AB 5E         3270          MOV   E,M      GET LOW BYTE OF DISK ADDR
19AC 34         3275          INR   M        UPDATE DISK ADDR FOR NEXT
19AD 23         3280          INX   H        DISK COMMAND
19AE 56         3285          MOV   D,M      GET HIGH BYTE OF DISK ADDR
19AF C2 B3 19   3290          JNZ   $+1
19B2 34         3295          INR   M        UPDATE HIGH BYTE IF NEEDED
19B3 D5         3300          PUSH  D        SAVE DISK ADDRESS
19B4 EB         3305          XCHG           D,E -> XXADR (HIGH BYTE)
19B5 CD 2D 19   3310          CALL  LOADHL   H,L <- FILE BLOCKS REMAINING
19B8 7C         3315          MOV   A,H      LOOK FOR ZERO BLOCKS REMAINING
19B9 B5         3320          ORA   L
19BA C2 C3 19   3325          JNZ   OKLEN    BRIF 1 OR MORE BLOCKS LEFT
19BD D1         3330          POP   D        CLEAR STACK
19BE D1         3335          POP   D
19BF D1         3340          POP   D
19C0 AF         3345          XRA   A        SET 'Z' FOR LENGTH ERROR
19C1 37         3350          STC            SET 'C' FLAG FOR ERROR
19C2 C9         3355          RET
19C3 2B         3360 OKLEN:   DCX   H        1 LESS BLOCK REMAINING
19C4 13         3365          INX   D        D,E -> XXLEN (LOW)
```

```
19C5 CD 26 19   3370          CALL  UPDATE   SAVE H,L -> (D,E)
19C8 E1         3375          POP   H        H,L -> DISK ADDRESS
19C9 D1         3380          POP   D        D,E -> START OF RAM BUFFER
19CA F1         3385          POP   PSW      RETRIEVE LENGTH
19CB CD 22 20   3390          CALL  DCOM     ISSUE DISK COMMAND
19CE D0         3395          RNC            RETURN IF NO ERROR
19CF 3E 81      3400          MVI   A,81H
19D1 87         3405          ADD   A        A=2 FOR DISK WRITE PROTECT
19D2 C9         3410          RET            'C' IS SET
19D3            3415 *
19D3            3420 * SEARCH DIRECTORY FOR FILENAME
19D3            3425 * H,L -> XXFILE (FILE NAME)
19D3            3430 * D,E -> XXUNIT (DISK UNIT #)
19D3            3435 * RETURN WITH 'C' SET IF FILENAME NOT FOUND
19D3            3436 * RETURN WITH 'Z' SET IF BAD FILENAME
19D3            3440 * RETURN WITH H,L->XXUNIT; D,E->DIR ENTRY OR DISK ADDR
19D3            3445 * SET XXUNIT TO DISK DRIVE #; 'M' FLAG IF FILE EXISTS
19D3            3450 *
19D3 3E 01      3455 SEARCH:  MVI   A,1      DEFAULT TO DISK DRIVE #1
19D5 CD D7 1A   3460          CALL  DIRLOOK
19D8 EB         3465          XCHG           D,E -> DIR ENTRY OR DISK ADDR
19D9 77         3470          MOV   M,A      H,L -> XXUNIT; SAVE UNIT #
19DA D8         3475          RC             RETURN IF FILE NAME NOT FOUND
19DB C8         3476          RZ             RETURN IF BAD FILENAME
19DC F6 80      3480          ORI   80H      SET 'M' FLAG IF FILE NAME FOUND
19DE 77         3485          MOV   M,A      DON'T CHANGE DIR LENGTH ON CLOSE
19DF            3490 *
19DF            3495 * MOVE DISK ADDR AND FILE LENGTH FROM DIRECTORY ENTRY
19DF            3500 * TO FILE TABLE ENTRY.
19DF            3505 * H,L -> XXUNIT; D,E -> DIRECTORY ENTRY
19DF            3510 * RETURN WITH FILE TYPE IN A-REG.
19DF            3515 *
19DF E5         3520 SRCONT:  PUSH  H
19E0 23         3525          INX   H        POINT TO LOW BYTE OF RA! 3UFFER
19E1 36 00      3530          MVI   M,0      START ON 100H BOUNDARY
19E3 23         3535          INX   H        POINT TO DISK ADDRESS
19E4 23         3540          INX   H
19E5 0E 04      3545          MVI   C,4      GET DISK ADDRESS AND FILE LENGTH
19E7 1A         3550 MOVDIR:  LDAX  D        FROM DIRECTORY AND MOVE
19E8 77         3555          MOV   M,A      TO XXADR AND XXLEN
19E9 13         3560          INX   D
19EA 23         3565          INX   H
19EB 0D         3570          DCR   C
19EC C2 E7 19   3575          JNZ   MOVDIR
19EF 1A         3580          LDAX  D        GET FILE TYPE FROM DIRECTORY
19F0 B7         3585          ORA   A        BE SURE 'C' FLAG IS CLEAR
19F1 D1         3590          POP   D        D,E -> XXUNIT
19F2 C9         3595          RET
19F3            3600 *
19F3            3605 * CREATE A DIRECTORY ENTRY FOR DISK FILE
19F3            3610 * A-REG CONTAINS FILE TYPE
19F3            3615 * H,L -> XXUNIT
19F3            3620 * D,E CONTAINS DISK ADDRESS FOR NEW FILE
19F3            3625 * B,C CONTAINS FILE LENGTH, IF 0 ALLOC ALL AVAIL SPACE
19F3            3630 * RETURN WITH 'C' SET IF ERROR
19F3            3635 * IF ERROR: A=0 ('Z' SET) IF DISK FULL
19F3            3636 *           A=1 ('Z' RESET) IF BAD FILE NAME
19F3            3637 *           A=2 ('Z' RESET) IF DISK WRITE PROTECT
19F3            3640 *
19F3 E5         3645 CREATE:  PUSH  H        SAVE FILE TABLE ADDRESS
19F4 F5         3650          PUSH  PSW      SAVE FILE TYPE
19F5 7E         3655          MOV   A,M      GET DISK DRIVE #
19F6 E6 03      3660          ANI   3        REMOVE POSSIBLE 'M' FLAG
19F8 21 10 1C   3665          LXI   H,BLANK  POINT TO BLANK "NAME"
19FB CD D7 1A   3670          CALL  DIRLOOK  LOCATE BLANK DIRECTORY ENTRY
19FE E5         3675          PUSH  H        SAVE POINTER TO DIRECTORY ENTRY
19FF DA 72 1A   3680          JC    DSKFULL  ERROR IF DIRECTORY IS FULL
1A02 73         3685          MOV   M,E      MOVE DISK ADDRESS TO DIRECTORY
1A03 23         3690          INX   H
1A04 72         3695          MOV   M,D
1A05 23         3700          INX   H        POINT TO LENGTH ENTRY
1A06 78         3705          MOV   A,B      SEE IF LENGTH IS ZERO
1A07 B1         3710          ORA   C
1A08 C2 1D 1A   3715          JNZ   USEBC    SET LENGTH TO CONTENTS OF B,C
1A0B 3E 5E      3720          MVI   A,94     COMPUTE FILE LENGTH AS
1A0D 93         3725          SUB   E        350-DISK ADDR (E.G. USE ALL
1A0E 5F         3730          MOV   E,A      AVAILABLE DISK SPACE FOR SPOOL)
1A0F 3E 01      3735          MVI   A,1
1A11 9A         3740          SBB   D
1A12 57         3745          MOV   D,A      D,E CONTAINS SPOOL FILE LENGTH
1A13 DA 72 1A   3750          JC    DSKFULL  ERROR IF DISK IS FULL
1A16 B3         3755          ORA   E        OR IF NEW FILE HAS 0 LENGTH
1A17 CA 72 1A   3760          JZ    DSKFULL
1A1A C3 2C 1A   3765          JMP   MOVLN
1A1D E5         3770 USEBC:   PUSH  H        SAVE POINTER TO LENGTH ENTRY
1A1E 21 A1 FE   3775          LXI   H,-351
1A21 09         3780          DAD   B        ADD LENGTH
1A22 DA 71 1A   3785          JC    DSKFL    LENGTH >350 BLOCKS
1A25 19         3790          DAD   D        ADD DISK ADDRESS
1A26 DA 71 1A   3795          JC    DSKFL    FILE TOO LONG
1A29 E1         3800          POP   H
1A2A 59         3805          MOV   E,C      USE LENGTH IN B,C
1A2B 50         3810          MOV   D,B
1A2C 73         3815 MOVLN:   MOV   M,E      MOVE LENGTH TO DIRECTORY
1A2D 23         3820          INX   H
1A2E 72         3825          MOV   M,D
1A2F 23         3830          INX   H        POINT TO TYPE
1A30 D1         3835          POP   D        D,E -> DIRECTORY ENTRY (BYTE 8)
1A31 F1         3840          POP   PSW      GET FILE TYPE
1A32 77         3845          MOV   M,A      SET FILE TYPE
1A33 EB         3850          XCHG           H,L -> DIRECTORY ENTRY
1A34 D1         3855          POP   D        D,E -> XXUNIT
1A35 D5         3860          PUSH  D
1A36 E5         3865          PUSH  H
1A37 01 F8 FF   3870          LXI   B,-8     MOVE SPOOL FILE NAME FROM XXFILE
1A3A 09         3875          DAD   B        TO DIRECTORY ENTRY AT H,L
1A3B EB         3880          XCHG           D,E -> START OF DIRECTORY ENTRY
1A3C 0E F4      3885          MVI   C,-12    SET B,C TO OFFSET OF XXFILE (-12)
1A3E 09         3890          DAD   B        H,L -> XXFILE
1A3F 0E 08      3895          MVI   C,8      COUNT FOR FILE NAME LENGTH
1A41 7E         3900 MOVNM:   MOV   A,M      MOVE NAME
1A42 FE 2C      3905          CPI   ','      STOP ON ',' DELIMETER
1A44 CA 5B 1A   3910          JZ    WRDIR
1A47 FE 0D      3915          CPI   CR       STOP ON CR
1A49 CA 5B 1A   3920          JZ    WRDIR
1A4C FE 20      3925          CPI   ' '      STOP ON BLANK
1A4E CA 5B 1A   3930          JZ    WRDIR
1A51 DA 79 1A   3935          JC    DSKNAME  KILL IF BAD CHAR
1A54 12         3940          STAX  D
1A55 23         3945          INX   H        NEXT CHAR
1A56 13         3950          INX   D
1A57 0D         3955          DCR   C        CHECK COUNT
1A58 C2 41 1A   3960          JNZ   MOVNM
1A5B 2A 2D 20   3961 WRDIR:   LHLD  DOSERR+1 GET ^C TV
1A5E E5         3962          PUSH  H
1A5F 21 80 1A   3963          LXI   H,DSKWP  SET UP TV FOR DISK
1A62 22 2D 20   3964          SHLD  DOSERR+1 WRITE PROTECT
1A65 CD 1F 20   3965          CALL  DWRIT    WRITE DIRECTORY TO DISK
1A68 E1         3966          POP   H        RESTORE ORIGINAL TV
1A69 22 2D 20   3967          SHLD  DOSERR+1
1A6C D1         3970          POP   D        D,E -> DIRECTORY ENTRY
1A6D E1         3975          POP   H        H,L -> XXUNIT
1A6E C3 DF 19   3980          JMP   SRCONT   MOVE DIR TO XXADR AND XXLEN
1A71 E1         3985 DSKFL:   POP   H        RESTORE STACK
1A72 E1         3990 DSKFULL: POP   H        RESTORE STACK
1A73 E3         3995          XTHL           THROW AWAY FILE TYPE
```

```
1A74 AF                   XRA   A        SET 'Z' FLAG
1A75 37                   STC            INDICATE ERROR
1A76 E1                   POP   H
1A77 D1                   POP   D        H,L -> DIR ENTRY; D,E -> XXUNIT
1A78 C9                   RET
1A79 3E 01      DSKNAME:  MVI   A,1      SEND ERROR CODE
1A7B B7                   ORA   A
1A7C 37                   STC
1A7D E1                   POP   H
1A7E D1                   POP   D
1A7F C9                   RET
1A80 E1         DSKWP:    POP   H        CLEAR RETURN ADDRESS
1A81 E1                   POP   H        GET ORIGINAL TV
1A82 22 2D 20             SHLD  DOSERR+1 RESTORE TV
1A85 3E 02                MVI   A,2      SEND ERROR CODE
1A87 C3 7B 1A             JMP   DSKNAME+2
1A8A            * CLOSE A FILE ON THE DISK
1A8A            * H,L -> WRFILE; D,E -> WRUNIT; B,C IS USED
1A8A            * A-REG CONTAINS DISK DRIVE #
1A8A            * 'M' FLAG SET IN WRUNIT MEANS DON'T CHANGE
1A8A            *     LENGTH ENTRY IN DIRECTORY
1A8A            * RETURN WITH 'C' FLAG IFF ERROR
1A8A            * IF ERROR: 'Z' SET FOR LENGTH ERROR, RESET FOR OTHER
1A8A            * IF ERROR: A=2 FOR WRITE PROTECT, A=4 FOR BAD FILENAME
1A8A            *
1A8A 06 00      CLSIT:    MVI   B,0      SET DISK WRITE COMMAND
1A8C 1A                   LDAX  D        GET DISK DRIVE #
1A8D E6 03                ANI   3        REMOVE POSSIBLE 'M' FLAG
1A8F C8                   RZ             RETURN IF FILE NOT OPEN
1A90 4F                   MOV   C,A
1A91 D5                   PUSH  D        SAVE REGS
1A92 E5                   PUSH  H
1A93 CD 2D 19             CALL  LOADHL   H,L -> RAM BUFFER
1A96 36 01                MVI   M,1      WRITE END-OF-FILE MARK
1A98 2E 00                MVI   L,0      H,L -> START OF BUFFER
1A9A EB                   XCHG
1A9B 23                   INX   H
1A9C 23                   INX   H        H,L -> WRBUF
1A9D 3E 01                MVI   A,1      WRITE 1 BLOCK
1A9F CD A8 19             CALL  DODSK    WRITE FINAL BLOCK TO DISK
1AA2 E1                   POP   H        H,L -> WRFILE
1AA3 D1                   POP   D        D,E -> WRUNIT
1AA4 D8                   RC             RETURN IF LENGTH ERROR
1AA5 1A                   LDAX  D        GET DISK UNIT #
1AA6 E6 03                ANI   3        REMOVE POSSIBLE 'M' FLAG
1AA8 CD D7 1A             CALL  DIRLOOK  FIND DIRECTORY ENTRY
1AAB D2 B2 1A             JNC   $+4
1AAE 3E 82                MVI   A,82H    BE SURE 'Z' IS RESET AND 'C' SET
1AB0 87                   ADD   A        A=4 FOR BAD FILENAME
1AB1 C9                   RET
1AB2 1A                   LDAX  D        GET DISK UNIT #
1AB3 B7                   ORA   A
1AB4 F5                   PUSH  PSW      SAVE 'M' FLAG
1AB5 AF                   XRA   A
1AB6 12                   STAX  D        ZERO DISK UNIT NUMBER
1AB7 13                   INX   D
1AB8 13                   INX   D
1AB9 13                   INX   D        D,E -> WRADR (CURR DISK ADDR)
1ABA 1A                   LDAX  D
1ABB 96                   SUB   M        H,L -> START DISK ADDR
1ABC 4F                   MOV   C,A      COMPUTE LENGTH OF DISK FILE
1ABD 13                   INX   D
1ABE 23                   INX   H
1ABF 1A                   LDAX  D
1AC0 9E                   SBB   M
1AC1 47                   MOV   B,A      B,C CONTAINS FILE LENGTH
1AC2 23                   INX   H        H,L -> DIR ENTRY FOR LENGTH
1AC3 F1                   POP   PSW      GET SAVE SIZE FLAG
1AC4 FA C8 1A             JM    $+1      DON'T CHANGE LENGTH IN
1AC7 71                   MOV   M,C      DIRECTORY IF 'M' FLAG
1AC8 23                   INX   H
1AC9 FA CD 1A             JM    $+1
1ACC 70                   MOV   M,B      UPDATE DISK FILE LENGTH
1ACD 23                   INX   H        SKIP TYPE BYTE
1ACE 23                   INX   H        H,L -> OTHER DIR INFO
1ACF 71                   MOV   M,C      ALWAYS SAVE ACTUAL LENGTH
1AD0 23                   INX   H        IN THIS FIELD IN DIRECTORY
1AD1 70                   MOV   M,B
1AD2 CD 1F 20             CALL  DWRIT    WRITE DIRECTORY TO DISK
1AD5 AF                   XRA   A        RESET ALL FLAGS
1AD6 C9                   RET
1AD7            *
1AD7            * LOOK UP FILENAME IN DISK DIRECTORY
1AD7            * H,L -> XXFILE; A-REG CONTAINS DRIVE #
1AD7            * RETURN WITH 'C' RESET IF FILENAME FOUND
1AD7            *                  H,L -> DIRECTORY ENTRY
1AD7            * RETURN WITH 'C' SET IF FILENAME NOT FOUND
1AD7            *                  H,L CONTAINS DISK ADDR
1AD7            * ON RETURN A-REG CONTAINS DISK DRIVE #
1AD7            *             IF A=0 ('Z' SET) THEN FILENAME IS BAD
1AD7            *
1AD7 C5         DIRLOOK:  PUSH  B        SAVE REGS
1AD8 D5                   PUSH  D
1AD9 E5                   PUSH  H
1ADA 2A 2D 20             LHLD  DOSERR+1 GET ERROR TV
1ADD 22 56 1C             SHLD  SAVERR   HOLD IT
1AE0 21 09 1B             LXI   H,FILNM  SET TV FOR BAD FILENAME
1AE3 22 2D 20             SHLD  DOSERR+1
1AE6 21 00 00             LXI   H,0      ALSO SAVE STACK POINTER
1AE9 39                   DAD   SP
1AEA 22 58 1C             SHLD  SAVSTK
1AED 2A B7 28             LHLD  DOSPTNR  SAVE DOS POINTER FOR DOS INPUT
1AF0 E3                   XTHL           ROUTINE WHICH MAY BE ACTIVE
1AF1 CD 1C 20             CALL  DLOOK    DOS DIRECTORY LOOK UP
1AF4 E3                   XTHL
1AF5 22 B7 28             SHLD  DOSPTNR  RESTORE DOS POINTER
1AF8 2A 56 1C             LHLD  SAVERR   RESTORE ERROR TV
1AFB 22 2D 20             SHLD  DOSERR+1
1AFE E1                   POP   H
1AFF D1                   POP   D
1B00 C1                   POP   B
1B01 DA 06 1B             JC    $+2
1B04 B7                   ORA   A        BE SURE 'Z' IS RESET
1B05 C9                   RET
1B06 B7                   ORA   A
1B07 37                   STC
1B08 C9                   RET
1B09 2A 58 1C   FILNM:    LHLD  SAVSTK   RESTORE STACK POINTER
1B0C F9                   SPHL
1B0D E1                   POP   H        RESTORE DOS POINTER
1B0E 22 B7 28             SHLD  DOSPTNR
1B11 D1                   POP   D        ADJUST STACK
1B12 C1                   POP   B
1B13 AF                   XRA   A        SET ZERO
1B14 37                   STC            ALSO FILENAME NOT FOUND
1B15 C9                   RET
1B16            *
1B16            * ROUTINES TO CONTROL DISK DRIVE MOTORS AND HEAD LOADING
1B16            *
1B16            * TURN ON MOTORS, IF OFF INDICATE NO DRIVE SELECTED
1B16            *
1B16 3A 90 EB   STRTM:    LDA   RESET    TURN ON MOTORS, READ A STATUS
1B19 E6 10                ANI   MO       LOOK AT MOTOR-ON BIT
1B1B C0                   RNZ            DONE IF MOTORS WERE ON
1B1C 3E 59                MVI   A,NOSEL  SET FLAG FOR NO DRIVE
1B1E 32 03 20             STA   DRVSEL   SELECTED
1B21 C9                   RET
1B22            *
1B22            * CHECK IF DISK DRIVE MOTORS ARE ON
1B22            * RESET MOTOR TIMER IFF ON
1B22            *
1B22 3A 10 EB   CKMOTOR:  LDA   READA    READ A STATUS BYTE
1B25 E6 10                ANI   MO       LOOK AT MOTOR-ON BIT
1B27 C8                   RZ             FORGET IT IF MOTORS OFF
1B28 3A 90 EB             LDA   RESET
1B2B C9                   RET
1B2C            *
1B2C            * SELECT DISK UNIT, LOAD HEAD
1B2C            * C <- DRIVE NUMBER; B <- CONTROLLER COMMAND
1B2C            *
1B2C 3A 03 20   SELECT:   LDA   DRVSEL
1B2F B9                   CMP   C        SEE IF DESIRED DRIVE IS
1B30 C8                   RZ             ALREADY SELECTED, DONE IF SO
1B31 0A                   LDAX  B        SELECT DRIVE
1B32 79                   MOV   A,C      UPDATE CURRENT DRIVE
1B33 32 03 20             STA   DRVSEL   SELECTED
1B36 C9                   RET
1B37            *
1B37            * ERROR ROUTINES FOR SPOOL PROBLEMS
1B37            *
1B37 21 86 1B   WRERR:    LXI   H,BADPAR PRINT MESSAGE FOR WRITE PROTECT
1B3A C3 3C 1C             JMP   STOP
1B3D 21 FE 1B   LNERR:    LXI   H,TOLONG PRINT MESSAGE FOR FILE OVERFLOW
1B40 C3 3C 1C             JMP   STOP     STOP SPOOL ACTIVITY
1B43 EB         RDERR:    XCHG           D,E POINTS TO RDUNIT
1B44 21 9D 1B             LXI   H,RDNF   PRINT MESSAGE FOR READ
1B47 C3 3C 1C             JMP   STOP     SPOOL NOT FOUND
1B4A 21 B4 1B   FLERR:    LXI   H,FULL   SET MSG FOR DISK FULL
1B4D C3 3C 1C             JMP   STOP
1B50 E1         NPERR:    POP   H        CLEAR RET ADDR (SEE SPLSRCH)
1B51 21 CC 1B   NMERR:    LXI   H,BADNM  SET MSG FOR BAD FILE NAME
1B54 C3 3C 1C             JMP   STOP
1B57 21 BE 1B   TYERR:    LXI   H,BADTYP PRINT MESSAGE FOR
1B5A C3 3C 1C             JMP   STOP     WRONG FILE TYPE ON SPOOL
1B5D F5         CLSERR:   PUSH  PSW      SAVE ERROR CODE
1B5E 21 F0 1B             LXI   H,BADCLS ERROR ON CLOSE
1B61 CD 2E 1C             CALL  PMSG
1B64 F1                   POP   PSW      GET ERROR CODE
1B65 21 B4 1B             LXI   H,FULL   DISK FILE FULL IF
1B68 CA 3C 1C             JZ    STOP     'Z' SET, ELSE
1B6B 21 A7 1B             LXI   H,NF     FILENAME NOT FOUND IF
1B6E FE 04                CPI   4        ERROR CODE IS 4
1B70 CA 3C 1C             JZ    STOP
1B73 21 86 1B             LXI   H,BADPAR ELSE DISK IS WRITE PROTECTED
1B76 C3 3C 1C             JMP   STOP     STOP SPOOL
1B79 21 18 1C   ESCERR:   LXI   H,BADESC
1B7C CD 2E 1C             CALL  PMSG
1B7F CD 4D 19             CALL  DSPBELL
1B82 3E 80                MVI   A,80H    SET ERROR BIT
1B84 B7                   ORA   A        AND FLAGS
1B85 C9                   RET
1B86            *
1B86            * ERROR MESSAGES FOR SPOOL PROBLEMS
1B86            *
1B86 0D         BADPAR:   DB    CR
1B87 0A                   DB    LF
1B88 44 49 53             ASC   'DISK WRITE-PROTECTED'
     4B 20 57 52 49 54 45 2D 50 52
     4F 54 45 43 54 45 44
1B9C 00                   DB    0
1B9D 52 45 41   RDNF:     ASC   'READ SPOOL'
     44 20 53 50 4F 4F 4C
1BA7 20 4E 4F   NF:       ASC   ' NOT FOUND'
     54 20 46 4F 55 4E 44
1BB1 0D                   DB    CR
1BB2 0A                   DB    LF
1BB3 00                   DB    0
1BB4 44 49 53   FULL:     ASC   'DISK FULL'
     4B 20 46 55 4C 4C
1BBD 00                   DB    0
1BBE 42 41 44   BADTYP:   ASC   'BAD FILE TYPE'
     20 46 49 4C 45 20 54 59 50 45
1BCB 00                   DB    0
1BCC 0D         BADNM:    DB    CR
1BCD 0A                   DB    LF
1BCE 42 41 44             ASC   'BAD FILE NAME'
     20 46 49 4C 45 20 4E 41 4D 45
1BDB 00                   DB    0
1BDC 53 50 4F   CLOSED:   ASC   'SPOOL FILE CLOSED'
     4F 4C 20 46 49 4C 45 20 43 4C
     4F 53 45 44
1BED 0D         CRLF:     DB    CR
1BEE 0A                   DB    LF
1BEF 00                   DB    0
1BF0 43 4C 4F   BADCLS:   ASC   'CLOSE ERROR -'
     53 45 20 45 52 52 4F 52 20 2D
1BFD 00                   DB    0
1BFE 0D         TOLONG:   DB    CR
1BFF 0A                   DB    LF
1C00 53 50 4F             ASC   'SPOOL FILE FULL'
     4F 4C 20 46 49 4C 45 20 46 55
     4C 4C
1C0F 00                   DB    0
1C10 20 20 20   BLANK:    ASC   '
     20 20 20 20 20
1C18 42 41 44   BADESC:   ASC   'BAD ESCAPE SEQUENCE'
     20 45 53 43 41 50 45 20 53 45
     51 55 45 4E 43 45
1C2B 0D                   DB    CR
1C2C 0A                   DB    LF
1C2D 00                   DB    0
1C2E            *
1C2E            * PRINT MESSAGE, STOP ON '0' OR NEGATIVE BYTE
1C2E            * H,L -> MESSAGE
1C2E            *
1C2E 7E         PMSG:     MOV   A,M      GET CHAR
1C2F B7                   ORA   A        LOOK FOR END OF MESSAGE
1C30 C8                   RZ
1C31 F8                   RM
1C32 CD 4F 19             CALL  DISPLAY  PRINT MESSAGE
1C35 23                   INX   H
1C36 C3 2E 1C             JMP   PMSG
1C39            *
1C39            * CLOSE READ SPOOL WHEN '1' BYTE FOUND
1C39 21 DC 1B   STOPR:    LXI   H,CLOSED "CLOSED" MESSAGE
1C3C            *
1C3C            * PROCESS ERROR RETURN ON READ/WRITE
1C3C            * CLOSE ALL SPOOL FILES FOR ERROR CONDITION
1C3C            * STACK -> IOFLAG; D,E -> XXUNIT
1C3C            *
1C3C CD 2E 1C   STOP:     CALL  PMSG     PRINT MESSAGE
1C3F 21 ED 1B             LXI   H,CRLF   PRINT CR AND LF
1C42 CD 2E 1C             CALL  PMSG
1C45 E1                   POP   H        H,L -> IOFLAG
1C46 CD 4D 19             CALL  DSPBELL  SIGNAL ERROR OR END
1C49 AF                   XRA   A        TURN OFF ACTIVE SPOOL
1C4A 12                   STAX  D        D,E -> XXUNIT
1C4B 3E 80                MVI   A,80H    SET ERROR BIT IN IOFLAG
1C4D B6                   ORA   M
1C4E 77                   MOV   M,A
1C4F C9                   RET            RETURN TO DOS I/O ROUTINE
1C50            *                        WITH 'M' FLAG SET
1C50            *
1C50            *STORAGE SPACE FOR I/O ROUTINES
```

```
1C50                5135 *
1C50 00 00          5140 RDSTK    DW     0          POINTER TO COMMAND STACK
1C52 95 1C          5145 STACK    DW     LOCSTK     ADDRESS OF STACK AREA
1C54 00             5150 IOFLAG   DB     0          INPUT/OUTPUT DEVICE CONTROL FLAG
1C55 00             5155 IODEV    DB     0          SELECTED I/O DEVICE
1C56 00 00          5156 SAVERR   DW     0          DOS ERROR TV AT DOSERR
1C58 00 00          5157 SAVSTK   DW     0          HOLD STACK POINTER
1C5A                5160 *
1C5A                5165 *TABLES FOR DISK I/O SPOOL ROUTINES
1C5A                5170 *
1C5A                5175          DS     2          SPACE FOR POINTER IN NAME CHNG
1C5C                5180 SPLFILE  EQU    $          START OF TABLE FOR SPOOL FILES
1C5C                5185 *WRITE SPOOL
1C5C 53 50 4F       5190 WRFILE   ASC    'SPOOL,2   '  WRITE SPOOL FILE NAME
     4F 4C 2C 32 20 20 20
1C66 0D             5195          DB     13         PROPER TERMINATION
1C67 FF             5200          DB     0FFH       TERM NAME INPUT
1C68 00             5205          DB     0          ACTIVE DRIVE #
1C69 00 1D          5210          DW     SPLBUF     POINTER TO RAM BUFFER
1C6B 00 00          5215          DW     0          CURRENT DISK ADDRESS
1C6D 00 00          5220          DW     0          WRITE SPOOL FILE LENGTH
1C6F                5225 *READ SPOOL
1C6F 53 50 4F       5230 RDFILE   ASC    'SPOOL,2   '  READ SPOOL FILE NAME
     4F 4C 2C 32 20 20 20
1C79 0D             5235          DB     13         PROPER TERMINATION OF NAME
1C7A FF             5240          DB     0FFH       TERM NAME INPUT
1C7B 00             5245 RDUNIT   DB     0          ACTIVE DRIVE # (0 IF SPOOL OFF)
1C7C 00 1E          5250          DW     SPLBUF+100H  POINTER TO RAM BUFFER
1C7E 00 00          5255          DW     0          CURRENT DISK ADDRESS
1C80 00 00          5260          DW     0          READ SPOOL FILE LENGTH
1C82                5265 *PRINT SPOOL
1C82 50 52 49       5270 PRFILE   ASC    'PRINT,2   '  PRINT SPOOL FILE NAME
     4E 54 2C 32 20 20 20
1C8C 0D             5275          DB     13         PROPER TERMINATION
1C8D FF             5280          DB     0FFH       TERM NAME INPUT
1C8E 00             5285 PRUNIT   DB     0          ACTIVE DRIVE #
1C8F 00 1F          5290          DW     SPLBUF+200H  POINTER TO RAM BUFFER
1C91 00 00          5295          DW     0          CURRENT DISK ADDRESS
1C93 00 00          5300          DW     0          PRINT SPOOL FILE LENGTH
1C95                5305 *
1C95                5310 * SPACE FOR COMMAND STACK
1C95                5315 *
1C95                5320 LOCSTK   EQU    $
1C95                5325          DS     80         USE 80 BYTES FOR COMMAND STACK
1CE5 FF             5330          DB     0FFH       COMMAND STACK
1CE6                5335 *
1CE6                5340 * SPACE FOR 8080 STACK WHEN IN I/O ROUTINES
1CE6                5345 *
1CE6                5350 IOSTK    EQU    $+25
-EOF-

R;

.t s100spl rcr
```

PRESS ON

Nothing in the world
can take the place
of persistence.
Talent will not;
Nothing is more common
than unsuccessful men
with talent.
Genius will not;
Unrewarded genius
is almost a proverb.
Education will not;
The world is full
of educated derelicts.
Persistence and determination alone
are omnipotent.

—Unknown

A MONITOR PROGRAM IN PASCAL

Jon Bondy

When I first started using UCSD Pascal, I frequently found myself wanting access to the hardware from Pascal, so that I could read ports or write to memory. Whenever I needed to do these kinds of things, I found myself booting up CPM or using a ROM monitor. Each time that I added an I/O board to the system, I wanted to try it out from Pascal, but I couldn't do it easily. It seemed to me that it wouldn't be that hard to write a Pascal program to act as my "eyes" into the machine's hardware, allowing me to watch peripherals, or even the Pascal operating system, from within a Pascal program.

The first step in doing this was to write four assembly language routines which could be called from Pascal to allow me to read and write memory and ports. They proved to be somewhat more difficult to write than I had anticipated, since the UCSD Version 1.5 documentation had a minor bug in it, but after some playing around, I wound up with the following routines.

```
.FUNC   MEMREAD,1       ;PARAM IS ADDRESS
POP     IX              ;RETURN ADDRESS
POP     IY              ;POP TWO WORDS OF ZEROS
POP     IY              ;POP TWO WORDS OF ZEROS
POP     HL              ;READ ADDRESS
LD      E,(HL)          ;READ BYTE
LD      D,0
PUSH    DE
JP      (IX)

.PROC   MEMWRITE,2      ;PARAMS ARE ADDRESS, DATA
POP     IX              ;RETURN ADDRESS
POP     DE              ;DATA
POP     HL              ;WRITE ADDRESS
LD      (HL),E          ;WRITE BYTE
JP      (IX)

.FUNC   PORTREAD,1      ;PARAM IS PORT NUMBER
POP     IX              ;RETURN ADDRESS
POP     IY              ;POP TWO WORDS OF ZEROS
POP     IY              ;POP TWO WORDS OF ZEROS
POP     BC              ;PORT NUMBER
IN      E,(C)           ;READ PORT
LD      D,0
PUSH    DE
JP      (IX)

.PROC   PORTWRITE,2     ;PARAMS ARE PORT, DATA
POP     IX              ;RETURN ADDRESS
POP     DE              ;DATA
POP     BC              ;PORT NUMBER
OUT     (C),E           ;WRITE PORT
JP      (IX)
```

The MEMREAD and PORTREAD functions have just one parameter, that begin the address (or port) to be read. They return the values read as the values of their functions. The MEMWRITE and PORTWRITE procedures require two parameters, the address (or port) to be written to, and the data to be sent to that address (or port).

The PROCEDUREs are quite simple. UCSD Pascal sets the stack up so that when an assembly routine is called, the top of the stack is the return address, just as with standard machine language subroutine calls. Since I have a Z80, I save that address in register IX for later use; if you have an 8080, you could save it in memory as demonstrated in the UCSD Pascal V1.5 Manual. The two parameters are the next items on the stack. They are pushed onto the stack in the order in which they appear in the procedure call, so they are popped off in reverse order. I then perform the memory or I/O operation required by the procedure and return.

The FUNCTIONs are slightly more complex, since a value must be returned to the UCSD Pascal calling routine. UCSD Pascal sets up the function return address at the top of the stack, just as with procedures, but underneath that it inserts two bytes of storage on the stack for the function value. This is for some sort of compatibility with the stack formats for standard (non-assembly language) UCSD Pascal functions. These words must be popped off of the stack in order to get at the parameter values. The documentation bug in Version 1.5 was that the manual stated that the function storage was pushed onto the stack BEFORE the parameters, while in fact it is exactly the reverse. Finally, when the memory or port read is complete, the resulting value must be pushed onto the stack before control is returned to the Pascal program. Although two words of memory are placed on the stack for the return value by the UCSD Pascal system, only one is sent back. This is because UCSD Pascal leaves enough room on the stack for a REAL variable to be returned (if required), but we are only sending back a single 16-bit word.

Once the routines were developed, I placed them in the Pascal Library using the LIBRARIAN program so that they would be loaded into my CODE files automatically. If you decide to use my program on your UCSD system, you must enter the assembly language routines as listed above, assemble them, and incorporate the resulting code files into your file SYSTEM. LIBRARY. You are then ready to deal with the Pascal program itself.

With these tools in hand, I set out to write a simple but useful monitor program in Pascal. I wanted to be able to dump memory in either hex or character format, modify memory, and read and write ports. As I liked the prompt lines which UCSD uses in their operating system, I decided to use a similar format for my program.

Jon Bondy, Box 148, Ardmore, PA 19003

"THE ORIGINAL"

Personal Computing® 80

®

Presents:

Personal Computing and Small Business Computer Show

The Largest Personal Computing Show in 1980

August 21, 22,23, 24th at the Philadelphia Civic Center

- **Major exhibits by the industries leading companies**
- **Thursday, Aug. 21st, Dealer Day ——— 12 Noon to 6 P.M.**
- **Friday and Saturday, Aug. 22, 23rd —— 9 A.M. to 6 P.M.**
- **Sunday, Aug. 24th ———————— 10 A.M. to 5 P.M.**
- **Free Seminars ● Robotics Contest ● Antique Computer Display**
- **Special Seminars and Tutorials about Computer Music,** Saturday, Aug. 23rd
- **3rd Annual Computer Music Festival,** Saturday Evening, Aug. 23rd
  (Computer Music Festival is sponsored by the Philadelphia Area Computer Society-Tickets on sale at show)
- **Computer Visual Arts Festival,** Sunday, Aug. 24th

**Advanced Registration
Saves Time & Money**

□ Send______Dealer-Retailer (4 days) Registrations at $10. each, $12. at door for Thursday-Sunday, Aug. 21, 22, 23, 24

□ Send______Regular Registrations (3 days) at $8. each, $10. at door for Friday-Sunday, Aug. 22, 23, 24 only.

Advanced Registrations will be mailed late July - early August. No Advanced Registrations accepted after Aug. 8th.

□ Send Exhibitor information or Phone 609-653-1188

COMPANY NAME ______________________________

NAME______________________________________

STREET ___________________________________

CITY ____________________STATE__________ZIP________

PHONE ____________________________________

Send To:

PERSONAL COMPUTING 80

Rt. 1, Box 242, Warf Rd., ● Mays Landing, NJ 08330

Monitor Program, cont'd...

UCSD Pascal offers a useful feature when one uses their character strings during input, since character and line editing may be performed any time up until you hit the carriage return to end the line. I decided to have most of my input as character strings in order to allow the user to correct mistakes easily. This meant that I needed a routine which accepted a character string as input, and interpreted the characters in the string as hex digits for me. That wasn't difficult to do, but for the memory dumps, I wanted to input both the starting and ending addresses as parts of the same character string. A naive hex conversion routine would cause problems, since it would be passed the same string twice and would return the starting address as the interpreted value both times. I modified my routine to change the characters which it processed into blanks, so that the second call to the routine would skip over the (now blank) first field. The routine in its final form is given below:

```
function hexread(var chars : string) : integer;
  { convert the character string 'chars' into an integer by
      interpreting
   the characters in the string as hex digits }
 var
   temp : integer;
   done : boolean;
   i : integer;
 begin
 temp := 0;
 i := 1;
 done := false;
 { scan string until a non-blank is found }
 repeat
   if (i <= length(chars)) then
     if (chars[i] = ' ') then
       i := i + 1
     else done := true { a non-blank character was found }
   else done := true { scan has proceeded beyond end of string }
   until done;
 done := false;
 { add hex digits to the number being generated ('temp') until a
   non-hex digit is encountered }
 repeat
   if (i <= length(chars)) then
     if (chars[i] in ['0'..'9','A'..'F','a'..'f']) then begin
       if (chars[i] < 'A') then { it is a numeric digit }
         temp := (temp * 16) + ord(chars[i]) - ord('0')
       else begin
         if (chars[i] > 'F') then { convert lower case alpha
               to upper }
           chars[i] := chr(ord(chars[i]) - 32);
         { add all alpha's into number being generated }
         temp := (temp * 16) + ord(chars[i]) - ord('A') + 10;
         end; { else }
       { put blank in character processed so that if the same
          string
         is processed again,
        the first field will be skipped over }
    chars[i] := ' ';
    i := i + 1;
    end { if }
  else done := true { a non-hex character has been encountered }
  else done := true { the scan has proceeded beyond the
    end of string
     until done;
 { return number acquired }
 hexread := temp
 end; { hexread }
```

I wanted nicely formatted dumps, so I paid attention to the placement of the dumped data on the screen. If the user asked for a memory dump starting at address 0CH, I wanted the first byte dumped to be placed under the header line as the twelfth position on the line, rather than the first. In this way, one could always use the header line to determine which value was which. For example, a dump from 0CH to 12H would look like the following:

```
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

0000                                     03 02 EF F3
0010 C3 00 34
```

To do this, I calculated the number of spaces required after the address field ('0000' in the above example) in the variable 'numspaces', and printed them as WRITE (' ':numspaces).

For the memory modification routine, I copied a method which I have seen in so many places that I have forgotten where I first saw it. The user enters an address, after which the computer displays the contents of that byte in hex. The user can enter a space, a carriage return, or a hex value. If a space is entered, the value of the next byte in memory is displayed, and the process continues. If a carriage return is entered, the routine terminates. If a hex value is entered, that value is used to replace the value which was just displayed, and the value of the next memory location is displayed. Notice that since I was dealing with input data a character at a time in this routine, I could not use the 'hexread' routine which was discussed above. The very flexibility which the strings offered me in the first place (character and line editing) made it impossible to use them when I wanted real time interactive user input.

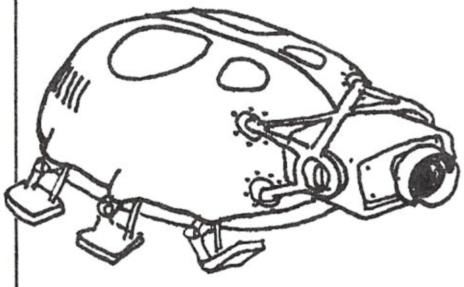
All in all, it took me about an hour to code the original version of the program. I quickly discovered that I needed a way to terminate the dumps if they were not of interest, and to freeze them if they were. I added a routine which read the keyboard, and used that to cause a screen freeze if a <control-s> was entered and dump termination if a <control-o> was entered. Note that I could not use standard Pascal I/O for this purpose, since a READ statement would have waited until the user entered a character rather than simply sampling the keyboard from time to time. The keyboard read routine is given below:

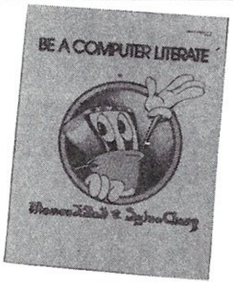
```
function kbd : char;
  { read the keyboard port, whether there is a new character
   there or not, and return that character to the caller }
  var
    i : integer;
  begin
  i := portread(1);
  { if msb is set, clear it }
  if (i < 128) then kbd := chr(i)
  else kbd := chr(i - 128)
  end; { kbd }
```

The final program is given below. I have transported it to a number of other UCSD systems with little difficulty, so I expect that you could enter it and use it without a problem.

```
 program monitor;

const
  eeos = 22;
  eeol = 23;
  home = 21;
  cntls = 19;
  cntlo = 15;

var
  port, addr, addr1, addr2, data : integer;
  chars : string;
  ch : char;
  goodchars : set of char;

procedure memwrite(addr, data : integer); external;
function memread(addr : integer) : integer; external;
procedure portwrite(port, data : integer); external;
function portread(port : integer) : integer; external;

function kbd : char;
  { read the keyboard port, whether there is a
     new character there or not,
  and return that character to the caller }
var
  i : integer;
begin
```

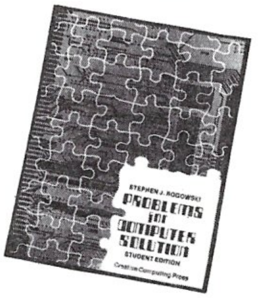
Have You Been Bitten By The Computer Bug?





Be A Computer Literate

A first introduction to the computer world for children age 10 to 14. Full color diagrams, drawings, photos and large type make this book easy to read and understand. Written by Marion J. Ball and Sylvia Charp. This informative 62-page book is used in many school systems. Softbound [6H] $3.95.



Problems for Computer Solution

Problems for Computer Solution by Stephen J. Rogowski is an excellent source of exercises in research and problem solving for students and self-learners. Problems like the Faulty Speedometer Spotter make learning fun and easy. 104 pages, softbound, [9Z] $4.95.

Creative Computing Magazine

Creative Computing has long been Number 1 in applications and software for micros, minis, and time-sharing systems for homes, schools and small businesses. Loads of applications every issue: text editing, graphics, communications, artificial intelligence, simulations, data base and file systems, music synthesis, analog control. Complete programs with sample runs. Programming techniques: sort algorithms, file structures, shuffling, etc. Coverage of electronic and video games and other related consumer electronics products, too.

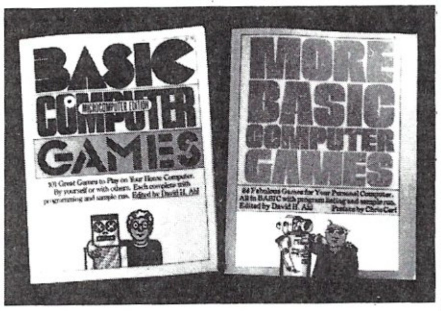
Just getting started? Then turn to our technology tutorials, learning activities, short programs, and problem solving pages. No-nonsense book reviews, too. Even some fiction and foolishness.

Subscriptions: 1 year $15, 3 years $40. Foreign, add $9/year surface postage, $26/year air.

Basic Computer Games

Edited by David Ahl, this book contains 101 imaginative and challenging games for one, two, or more players — Basketball, Craps, Gomoko, Blackjack, Even Wins, Super Star Trek, Bombs Away, Horserace. Simulate lunar landings. Play the stock market. Write poetry. Draw pictures.

All programs are complete with listing in Microsoft Basic, sample run and description. Basic conversion table included. 125,000 copies in print. 192 pages softbound. [6C] $7.50.



More Basic Computer Games

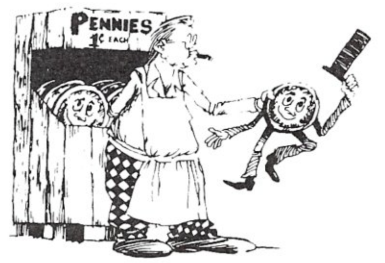
Contains 84 fascinating and entertaining games for solo and group play — evade a man-eating rabbit, crack a safe, tame a wild horse, become a millionaire, race your Ferrari, joust with a knight, trek across the desert on your camel, navigate in deep space.

All games come complete with program listing in Microsoft Basic, sample run and description. 192 pages softbound. [6C2] $7.50.

The Best of creative computing
Volume 1 Edited by David H. Ahl
THE BEST OF creative computing
VOL. 2 EDITED BY DAVID AHL

The Best of Creative Computing

The first two years of **Creative Computing** magazine have been edited into two big blockbuster books. **American Vocational Journal** said of Volume 1, "This book is the 'Whole Earth Catalog' of computers." [6A] Volume 2 continues in the same tradition. "Non-technical in approach, its pages are filled with information, articles, games and activities. Fun layout." —**American Libraries.** [6B] Each volume $8.95.



Computer Coin Games

Computer Coin Games by Joe Weisbecker aids newcomers to the field of computers by simplifying the concepts of computer circuitry through games which can be played with a few pennies and full sized playing boards in the book. Enhanced by outrageous cartoons, teachers, students and self-learners of all ages will enjoy this 96 page softbound book. [10R] $3.95.

How To Order

Send order and payment to **Creative Computing**, P.O. Box 789-M, Morristown, NJ 07960. Add $1.00 shipping and handling per order (foreign, $2.50) N.J. residents add 5% sales tax. Visa, MasterCharge and American Express orders welcome. For faster service, call in your bank card order toll free to: **800-631-8112.** (In NJ, call (201) 540-0445.)

creative computing

Monitor Program, cont'd...

```
i := portread(1);
{ if msb is set, clear it }
if (i < 128) then kbd := chr(i)
else kbd := chr(i - 128)
end; { kbd }

function hexread(var chars : string) : integer;
  { convert the character string 'chars' into an integer by interpreting
    the characters in the string as hex digits }
  var
    temp : integer;
    done : boolean;
    i : integer;
  begin
  temp := 0;
  i := 1;
  done := false;
  { scan string until a non-blank is found }
  repeat
    if (i <= length(chars)) then
      if (chars[i] = ' ') then
        i := i + 1
      else done := true { a non-blank character was found }
    else done := true { scan has proceeded beyond end of string }
    until done;
  done := false;
  { add hex digits to the number being generated ('temp') until a
    non-hex digit is encountered }
  repeat
    if (i <= length(chars)) then
      if (chars[i] in ['0'..'9','A'..'F','a'..'f']) then begin
        if (chars[i] < 'A') then { it is a numeric digit }
          temp := (temp * 16) + ord(chars[i]) - ord('0')
        else begin
          if (chars[i] > 'F') then { convert lower case alpha to upper }
            chars[i] := chr(ord(chars[i]) - 32);
          { add all alpha's into number being generated }
          temp := (temp * 16) + ord(chars[i]) - ord('A') + 10;
          end; { else }
        { put blank in character processed so that if the same string
          is processed again, the first field will be skipped over }
        chars[i] := ' ';
        i := i + 1;
        end { if }
      else done := true { a non-hex character has been encountered }
    else done := true { the scan has proceeded beyond the end of string }
    until done;
  { return number acquired }
  hexread := temp
  end; { hexread }

procedure hexwrite(data : integer);
  { write the integer 'data' to the CONSOLE: in hex }
  var
    temp1, temp2 : integer;
  begin
  temp1 := (data div 16) mod 16;
  temp2 := data mod 16;
  if (temp1 > 9) then write(chr(ord('A') + temp1 - 10))
  else write(chr(ord('0') + temp1));
  if (temp2 > 9) then write(chr(ord('A') + temp2 - 10))
  else write(chr(ord('0') + temp2))
  end; { hexwrite }
procedure asciidump(addr1,addr2 : integer);
  { dump memory starting at 'addr1' and stopping at 'addr2' to
    the CONSOLE: as ASCII }
```

```
var
  addr, temp : integer;
begin
{ write header line }
writeln(
  '     0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF');
{ write starting address }
hexwrite(addr1 div 256);
hexwrite(addr1 mod 256);
{ position cursor correctly for initial write }
write(chr(eeol),' ':((addr1 mod 64)+1));
addr := addr1;
repeat
  temp := memread(addr);
  { only write printable characters to the screen }
  if (chr(temp) in goodchars) then write(chr(temp))
    else write('.');
  addr := addr + 1;
  { only write 64 characters on a line }
  if ((addr mod 64) = 0) then begin
    writeln;
    hexwrite(addr div 256);
    hexwrite(addr mod 256);
    write(' ')
    end;
  { pause if the user types in a <control-s>; continue if another
    character is entered }
  while (kbd = chr(cntls)) do begin end
  { continue printing until the address limit is reached, or the user
    enters a <control-o> }
  until (addr = addr2) or (kbd = chr(cntlo));
{ write final character }
temp := memread(addr);
if (chr(temp) in goodchars) then write(chr(temp))
else write('.');
writeln;
end; { asciidump }

procedure dispmem(addr1,addr2 : integer);
  { dump memory in hex to screen, from 'addr1' to 'addr2' }
  var
    addr : integer;
  begin
  { write header }
  writeln('     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F');
  { write initial address }
  hexwrite(addr1 div 256);
  hexwrite(addr1 mod 256);
  { position cursor for initial write }
  write(chr(eeol),' ':(((addr1 mod 16) * 3)+1));
  addr := addr1;
  repeat
    hexwrite(memread(addr));
    write(' ');
    addr := addr + 1;
    { only write 16 bytes per line }
    if ((addr mod 16) = 0) then begin
      writeln;
      hexwrite(addr div 256);
      hexwrite(addr mod 256);
      write(' ')
      end;
    { pause if the user types in a <control-s>; continue if another
      character is entered }
    while (kbd = chr(cntls)) do begin end
    { continue printing until the address limit is reached, or the user
      enters a <control-o> }
    until (addr = addr2) or (kbd = chr(cntlo));
  { write final byte to screen }
```

```
hexwrite(memread(addr));
writeln;
end; { dispmem }
procedure substitute(addr : integer);
  { display and replace byte values in memory starting at address 'addr'  }
  var
    temp : integer;
    ch : char;
  begin
  { write the address }
  hexwrite(addr div 256);
  hexwrite(addr mod 256);
  write(' ');
  if eoln then readln;
  { repeat until the user enters a <CR> }
  while not eoln do begin
    { write value in memory }
    hexwrite(memread(addr));
    write('-');
    temp := 0;
    read(ch);
    { if user enters ' ', then proceed to next memory address.  otherwise,
      accept a new hex value to replace that which was displayed }
    if (ch <> ' ') then begin { accumulate hex value in 'temp' }
      while (ch in ['0'..'9','A'..'F','a'..'f']) do begin
        if (ch < 'A') then
          temp := (temp * 16) + ord(ch) - ord('0')
        else begin
          if (ch > 'F') then
            ch := chr(ord(ch) - 32);
          temp := (temp * 16) + ord(ch) - ord('A') + 10;
          end; { else }
        read(ch)
        end; { while }
      { write hex value into memory to replace value displayed }
      memwrite(addr,temp)
      end; { ch <> ' ' }
    addr := addr + 1;
    { only process 8 bytes per line }
    if ((addr mod 8) = 0) then begin
      writeln;
      hexwrite(addr div 256);
      hexwrite(addr mod 256);
      write(' ');
      if eoln then readln
      end
    end { while }
  end; { substitute }
```

```
 begin { main }
 goodchars := [' '..'
'];
  repeat
    { display prompt line }
    write(chr(home),
      'Monitor : D)mp,A)scii,S)ub,R)port,W)port,Q)uit');
    { accept command input }
    repeat
      read(keyboard,ch)
      until (ch in ['d','a','s','r','w','q',
                    'D','A','S','R','W','Q']);
    writeln(chr(eeos));
    { dispatch control to correct routine }
    case ch of
      'd','D' : begin
        write('Dump (start, stop) : ');
        readln(chars);
        { repeated calls to 'hexread' cause subsequent hex fields to be
          interpreted and returned as the value of the function }
        dispmem(hexread(chars),hexread(chars))
        end;
      'a','A' : begin
        write('Ascii (start, stop) : ');
        readln(chars);
        asciidump(hexread(chars),hexread(chars))
        end;
      's','S' : begin
        write('Substitute (addr) : ');
        readln(chars);
        substitute(hexread(chars))
        end;
      'r','R' : begin
        write('Read port (port) : ');
        readln(chars);
        hexwrite(portread(hexread(chars)))
        end;
      'w','W' : begin
        write('Write port (port, data) : ');
        readln(chars);
        portwrite(hexread(chars),hexread(chars))
        end
      end { case }
    until (ch = 'q') or (ch = 'Q')
  end.
                          { file is 'monitor.text' }
```

S-100 PROCESSOR BOARDS & MANUFACTURERS

Compiled by
Sol Libes

The following listing does not pretend to be complete. In fact, I would welcome reader additions and corrections to the listing. I prepared the listing in response to many letters that I have received regarding the large variety of CPU boards that I had previously mentioned as being available to S-100 based systems. I had mentioned that there "were 11 different microprocessor CPU type boards for the S-100". When I compiled the listing I discovered 13 different microprocessors were implemented on S-100 CPU boards, from 31 different manufacturers. The list follows.

If anyone would like to undertake to present listings of other types of S-100 CPU boards (e.g. memory, I/O, disk controller, video, etc.) please contact me.

MICRO-PROCESSOR	CPU SUPPLIER
2650	Victoria Micro Digital,401 Dundee St, Victoria, TX 77901
6502	CGRS Microtech, POB 102, Langhorn, PA 19047
6802	Mic DaSys, 357 South Lorranie Blvd, Los Angeles, CA 90020
6809	Ackerman Digital, Suite 208, 110 York Rd, Elmhurst, IL 60126
8080	Electronic Control Technology, 763 Ramsey Ave, Hillside, NJ 07205
	IMSAI Computer Div., FischerFreitas Corp., 2175 Adams Ave, San Leandro, CA 94501
	SSM Microcomputer Products, 2190 Paragon Drive, San Jose, CA 95131
	Wamenco, Inc., POB 877, El Granada, CA 94018
8085	Godbout Electronics, Bldg 725, Oakland Airport, CA 94614
	Dynabyte, 115 Independence Dr., Menlo Park, CA 94025
	Artec Electronics Inc., 605 Old County Rd, San Carlos, CA 94070
8088	Godbout Electronics, Bldg 725, Oakland Airport, CA 94614
	Lomas Data Products, 11 Cross Street, Westborough, MA 01581
8086	Seattle Computer Products, 1114 Industry Dr., Seattle, WA
	TecMar Inc, 23414 Greenlawn, Cleveland, OH 44122
9900	Marinchip Systems, 16 Saint Jude Rd, Mill Valley, CA 94941
Z-80	California Computer Systems,250 Caribbean, Sunnyvale, CA 94086
	CMC Marketing, 10611 Harwin, Suite 406, Houston, TX 77036
	Cromemco Inc, 280 Bernardo Ave, Mountain View, CA 94940
	Delta Products, 15392 Assembly Lane, Unit A, Huntington Beach, CA 92649
	Digital Research Computers, POB 401565, Garland, TX 75040
	Ithaca InterSystems, POB 91, Ithaca, NY 15850
	North Star Computers, Inc.,1440 Fourth St, Berkely, CA 94710
	QT Computer Systems Inc., 15335 South Hawthorne Blvd, Lawndale, CA 90260
	Quasar Data Products, 25151 Mitchell Dr, No. Olmstead, OH 44070
	SD Systems, POB 28810B, Dallas, TX 75228
	SSM Microcomputer Products, 2190 Paragon Drive, San Jose, CA 95131
	Tarbell Electronics, 950 Dovlen Pl, Suite B, Carson, CA 90746
	ZS-Systems/Zobex Inc., POB 1847, San Diego, CA 92112
Z-8000	Ithaca Intersystems, POB 91, Ithaca, NY 15850
	Quasar Data Products, 25151 Mitchell Dr., No. Olmsted, OH 44070
LSI-11	Alpha-Micro, 17881 Sky Park North, Irvine, CA 92714
PASCAL Microengine	Digicomp Research Corp., Terrace Hill, Ithaca, NY 14850

Software Consultants Harken!

Do you do software customizing? Do you do software package installation? Do you provide turnkey systems? Do you run a programming shop? Full-time? Part-time?

Starting with a future issue **MICROSYSTEMS** will publish a directory of "software shops." Places where computer customers can go for software package customizing. Most purchases of computer software packages find that the package does not exactly meet their needs—customizing is required—sometimes this is minor and sometimes it is major. Our directory will provide low cost advertising for "software shops." If you wish to be listed write to MICROSYSTEMS, Box 1192, Mountainside, NJ 07902 for information.

FREE CATALOGS

□ **Software.** Lists 400 programs on 70 tapes and disks. For education, recreation, and personal use.

□ **Books.** Lists 100 books, games, records, prints, etc. for educational and personal users of small computers.

□ **Peripherals.** (ALF music synthesizer and VersaWriter for the Apple II).

Send 3 15¢ stamps for either catalog or 5 for both. Or send $2.00 for a sample issue of Creative Computing and both catalogs.

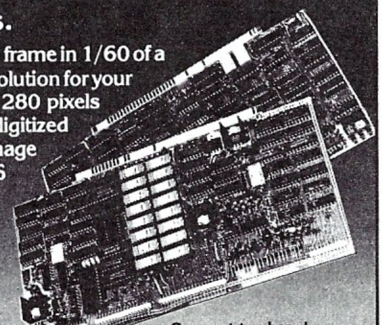
creative computing

P.O. Box 789-M
Morristown, NJ 07960

CAT-100 FULL COLOR GRAPHICS

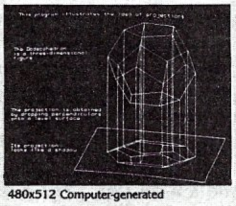
The original 256-color imaging system with high resolution video FRAME GRABBER for the S-100 bus.

Capture and digitize a video frame in 1/60 of a second. Select the best resolution for your application, from 256 to 1280 pixels per TV line. Display your digitized or computer processed image with 256 gray levels or 256 colors on standard B&W, NTSC or RGB color TV monitors.



Compact two-board basic system

240x256 Digitized image, 16 levels



480x512 Computer-generated

Features:

- Highest possible quality 480x512x8 digital video image presently available on the market
- Input capability from TV camera or other sources
- Variety of synchronization choices
- 2 selectable video A/D conversion circuits
- Choice of 1, 2, 4, 8, 16 or 32 bits per pixel
- 32K-byte image memory on the basic system
- 32, 64, 128 & 256K byte system capacity
- Lightpen input
- Photographic trigger control input
- Software selectable system parameters
- Interfaces for TRS-80 and other processors
- Comprehensive line of accessories, monitors and support software

SEND FOR FREE CATALOG

DIGITAL GRAPHIC SYSTEMS
441 California Ave., Palo Alto, CA 94306   415/494-6088

CATCH THE S-100 INC. BUS!

	LIST PRICE	OUR SPECIAL CASH PRICE
North Star Horizon II — 64K Double density A&T	3830.00	2999.00
Intertec's "Super Brain" W/32K, 2 drives, CP/M etc.	2995.00	2495.00
Godbout Econoram XIII — 24K Static RAM "Unkit"	479.00	399.00
S.D. Systems VDB 80X24 kit	370.00	*309.00
S.D. Systems SBC-100 Single board computer kit	295.00	*252.00

*Included free with every S.D. Systems board is an additional $25.00 manufacturer's rebate coupon.

Subject to Available Quantities • Prices Quoted Include Cash Discounts. Shipping & Insurance Extra.

We carry all major lines such as S.D. Systems, Cromemco, Ithaca Intersystems, North Star, Sanyo, ECT, TEI, Godbout, Thinker Toys, Hazeltine, IMC For a special cash price, telephone us.

Hours:
Mon.-Fri.
10 A.M.-6 P.M.

Bus .... **S-100, inc.**
Address ..... **7 White Place**
Clark, N.J. 07066
Interface **201-382-1318**

NEW PRODUCTS

RS-232-C TO RS-449 ADAPTERS

ITM Cables division announces a series of RS-232-C to RS-449 type data communication adapters. Adapters can be provided for RS-232 DTE to RS-449 DCE, or RD-449 DTE to RS-232 DCE. Each provides the proper electrical and mechanical interfaces between the "old" and the new EIA digital data systems. These adapters can ease the transition costs in computer and data communications. Interoperability is insured by compliance with EIA recommendations.

The RS-449 type devices include both RS-422 balanced and RS-423 unbalanced lines. Both primary (37 pin) and secondary (9 pin) channels can be supported with individual or combined adapters. A line of RS-449 cables is also available for primary only and combined circuits.

Loop back plugs provide a quick way of checking local equipment and cabling. Available for RS-232 and RS-423, these loop back the data and necessary control lines for local loopback testing. An LED indicator shows the presence and polarity of data traffic (line state). Catalog, selection sheets, and application guide available. Shipment from stock in most cases. OEM and quantity discounts. These adapters are available from: Interface Technology of Maryland, P.O. Box 745, College Park, MD 20740.

LOW-COST GRAPHICS PLOTTER

Strobe Inc. has just introduced a low cost, high performance digital graphics plotter. This new drum type plotter has a 0.004 inch step size, an 8 1/2 x 11 inch paper capacity, accepts a wide variety of pens, and uses quality stepping motor. In addition, this model features an interactive digitizing mode that allows the user to enter X-Y coordinate data corresponding to pen location directly into the host computer.

The plotter is controlled directly by the user's computer through two parallel output ports and one parallel input port. A hardware interface and software driver are available for S-100 bus machines. Also being offered is a plot software package providing vector generation and alphanumerics that runs with most versions of BASIC and FORTRAN.

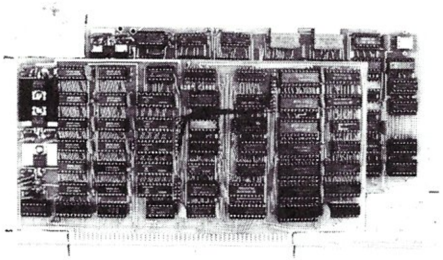
The price of the Model 100 Plotter is $680.00. Prices for software and hardware interfaces upon request. For further information contact STROBE INC., 897-5A Independence Ave., Mountain View, CA 94043.

32-CHANNEL DIGITAL SYNTHESIZER

A S-100 based digital synthesizer has been developed by Casheab. The device consists of two S-100 cards: a synthesizer card and a controller card. The synthesizer card is responsible for generating the music waveforms. All parameters are loaded into the synthesizer from the host processor. The timbre waveform is specified by 1024 12-bit samples. The synthesizer can hold up to 16 waveforms. The waveform selection for each of the 32 channels is specified by the processor. Frequency is specified as two bytes and amplitude is specified as one byte. The synthesizer is also capable of frequency modulation in which one channel frequency modulates a second channel. The host processor therefore has control over frequency, waveform, amplitude, and frequency modulation of each channel. The synthesizer is capable of additive synthesis, FM synthesis and direct digital synthesis.

The controller card is responsible for controlling the synthesizer card, summing the channel waveforms and handing the digital-to-analog conversion. The A-to-D conversion system consists of a 12-bit multiplying DAC and a 4-bit DAC. The 4-bit DAC is used to supply a reference voltage for the 12-bit DAC. This produces a greater dynamic range from the DAC.

Software on a CP/M compatible floppy disk is provided with the synthesizer. The software consists of a waveform creation program, a score compiling program and a play program. The waveform program is written in Basic and executes a frequency to time 1024 point FFT algorithm. Scores are written into Basic program using DATA statements; the program is compiled and the piece is played by using the play program.

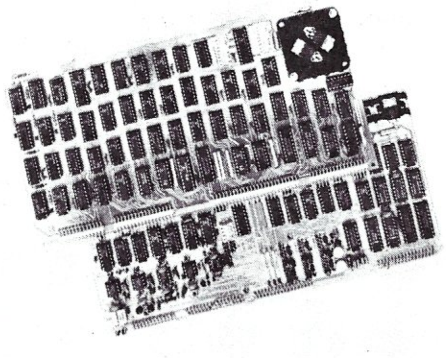


The 32-channel, 16 waveform synthesizer card (SYN-10/16) and the controller card (CTR-10) sell at a combined price of $1245.00. A 32-channel, 4-waveform (SYN-10/4) synthesizer and controller is also available at $1095.00. The manual for the synthesizer and controller is $5.00.

CASHEAB, 5737 Avenida Sanchez, San Diego, CA 92124. (714) 277-2547.

CROMEMCO INTRODUCES GRAPHICS INTERFACE

The Cromemco SDI is a high-resolution graphics interface designed for use in Cromemco computer systems. The SDI, displays color or black-and-white images with up to 756 by 484 point resolution, and with features such as high point resolution, color map selection, dual page windowing function, automatic area fill mode, and NTSC broadcast compatibility.



The SDI consists of two circuit boards which plug directly into the S-100 bus of any Cromemco microcomputer system and uses direct memory access to display the contents of a display memory. Each pixel of the display may be mapped from one nybble or from one bit of the display memory. Bit-mapped or nybble-mapped mode is software selectable. In fact, one part of the picture may be displayed in one mode and another part in the other mode. Also, 12K or 48K of memory may be used for the display memory, leading to four basic modes of operation.

The SDI has three separate analog output signals to drive the Red, Green and Blue guns of a high-quality RGB monitor, to preserve the full resolution of the picture. For black-and-white work any of these outputs can be used to drive black-and-white monitors. In fact, all three outputs can be used to display three totally different pictures to three different black-and-white monitors simultaneously. A composite SYNC signal is switch-selectable on any of these three outputs. A separate SYNC signal is also available.

These SYNC signals adhere strictly to the RS-170 standard for the television broadcast industry. In addition to the SYNC signal, the SDI outputs all signals required to serve as input to a colorizer or color modulator in a television broadcast studio. Furthermore, the SDI can be synchronized to external television equipment through the use of an external composite RS-170 SYNC signal, a composite video signal, or external horizontal and vertical SYNC signals applied to the appropriate SDI inputs.

Cromemco has also developed new 16K and 48K two-port memory cards for use with the SDI. Picture information is accessed by the SDI through a connector on the top of these cards. Use of the two-port memory for the display memory assures 75% to 100% CPU utilization, depending on the application software.

The two-board SDI graphics interface (Model SDI) is available for $595. For additional information, contact Cromemco, Inc., 280 Bernardo Avenue, Mountain View, CA 94043; (415) 964-7400.

AC POWER LINE ISOLATOR

Severe AC power line spikes, surges, noises and hash are prevalent in many MicroProcessor installations. Operators are plagued with unexplained crashes, memory loss and program glitches. Disks, printer and processor often interact, aggravating the problem.

ELECTRONIC SPECIALISTS' recently announced SUPER ISOLATOR is designed to curb these severe electrical problems. Incorporationg heavy duty surge/spike suppressors, the SUPER ISOLATOR features 3 individually dual-Pi filtered 3-prong AC sockets. Equipment interactions are eliminated and disruptive/damaging line spikes and hash are controlled. The SUPER ISOLATOR can accomodate an 1875 watt load, with each socket capable of handling 1000 watts.

Severe power line spike/surge and hash control is combined with interaction-free operation for $88.95 (Model ISO-3). SUPER ISOLATOR is available from: Electronic Specialists, Inc., 171 South Main Street, Natick, Massachusetts 01760. (617) 655-1532

In each issue of S-100 MICROSYSTEMS we will have this catalog listing of S-100 system software. If you have a software package you are offering for sale and want to bε listed then send us the information in the format shown. All information must be included. We reserve the right to edit and/or reject any submission.

SOFTWARE DIRECTORY

Program Name: VDRAW ASM
Hardware System: Any memory mapped video board with 2 x 3 Graphics: Polymorphic/IMSAI VIO/Vector G. Flashwriter
Minimum Memory Size: 1/4 K
Language: 8080 Assembler
Description: These routines will control memory-mapped video boards providing graphic capabilities. They will select and turn on or off any pixel desired. The user provides only an X and Y co-ordinate specifying the desired pixel for plot, or the X-Y co-ordinates of the start and end of a line. The routines will locate and set (or reset) the desired pixel or pixels. This will provide a simple interface for graphics from higher level languages. The plot routine will operate at very high speed. The draw routine, which utilizes the plot routine to set each pixel required, will draw a line on a video board so rapidly that the user will be unable to detect the time difference between the first and last pixels being set (or reset). The routine assumes that each pixel is controlled by a bit in an area occupied by a memory-mapped video board. The bits (pixels) must be arranged in a 2 x 3 matrix within a given byte (character) on the board. The two routines together will fit less than 256 bytes. The routines are also provided with two different methods for providing the X-Y addressing parameters. The parameters may be provided on the stack, or simply set into specified addresses.
Release: Currently available
Price: $30.00
Included with price: Program source code and documentation plus test program written in Basic.
Author: Hawkeye Grafix
Where to purchase it:

Hawkeye Grafix
23914 Mobile St.
Canoga Park, CA 91307

Program Name: COMM 4
Hardware System: CP/M and RS-232 Serial Port with modem.
Minimum Memory Size: 16K
Language: 8080 assembler
Description: Provides a comprehensive menu-driven communications package for users of CP/M operating systems linking to time-sharing or other CP/M systems. Terminal mode supports disk log option. Four file transfer modes perform auto disk paging without data loss, CRC-16 error retransmit, FDX no echo wait option, port-port/FDX/HDX modem. Local functions enable disk DIR, read name, delete, log in, plus control character and console echo switch.
Release: Now
Price: $150 source; $75 object
Included with price: Program and documentation.
Author: Hawkeye Grafix
Where to purchase it:

Hawkeye Grafix
23914 Mobile St.
Canoga Park, CA 91307

Program Name: Layout
Hardware System: Sol/Helios II, Disks (1)
Minimum Memory Size: 16K
Language: Extended Disk BASIC
Description: Layout saves programming time and effort by formatting, printing, and screen-printing a series of Data File Layouts. File produces uniform header for program description, and an indefinite number of descriptions of variables used in the data file. Provides space for programmer's comments. Excellent programming and reference tool.
Release: Now
Price: $20, includes source on disk and documentation.
Author: J. Brockway

Where to purchase it:

Jerry Brockway
Suite 308, 2909 Bay to Bay
Tampa, FL 33609.

Program Name: WHATSIT? (Wow! How'd All That Stuff get In There?) [WHATSIT? is a trademark of Computer Headware]
Hardware System: Any S-100 system; WHATSIT is available in Model NS-3 for North Star systems, and Model CP-2 for CP/M systems.
Minimum Memory Size: 32K (Model NS-3), 44K (Model CP-2).
Language: North Star BASIC (Model NS-3), CBASIC-2 (Model CP-2).
Description: WHATSIT is a self-indexing, cross referencing data query system. The program stores, indexes, and fetches free-format information in response to conversational "Requests." Typical queries range from "When's Johnny's Dental Checkup?" to "What's the U.N. Ambassador's Voting Record?" WHATSIT's unique open-ended data structure evolves continuously during normal use, without respecifying the file. Unexpected new file headings are immediately added when first mentioned in a Request, then remain available for future reference. Always spoken of as "her" in the 160-page user's manual, WHATSIT distinguishes herself by her breezy, impertinent repartee, including such rejoinders as "News to me!" when queried for information not currently on file, or "Never mind!" when the operator cancels a Request unexpectedly.
Release: March 1978 (Model NS-3), August 1979 (Model CP-2).
Price: $125.00 (Model NS-3), $175.00 (Model CP-2).
Included with price: Disk with 160-page spiral bound user's manual.
Author: Computer Headware, Box 14694, San Francisco, CA 94114.
Where to purchase it:

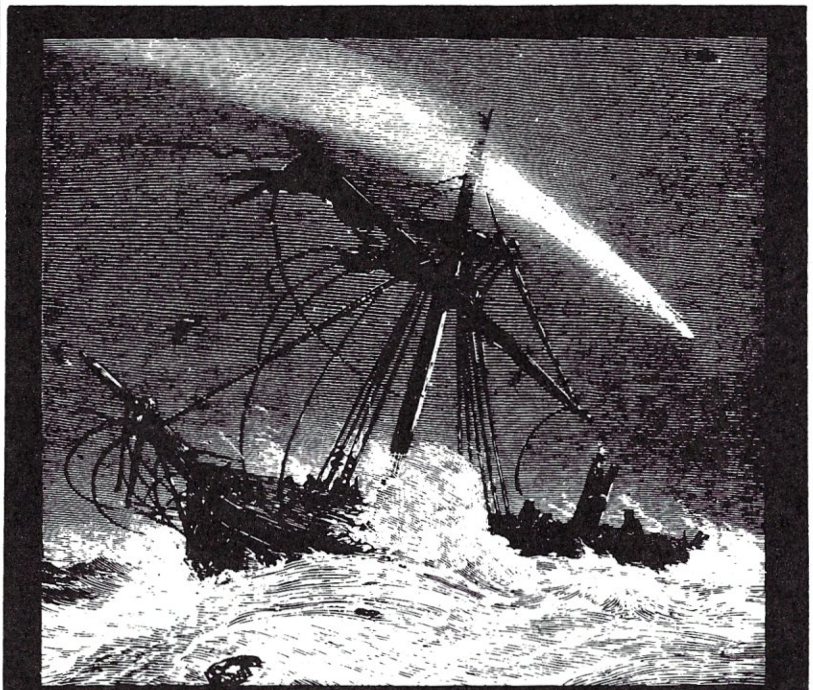
Hardhat Software
Box 14815
San Francisco, CA 94114

Program Name: muLISP-79
Hardware System: Standard CP/M
Minimum Memory Size: 20K
Language: LISP language interpreter
Description: Five man-years in the making and extensively tested, the muLISP-79 Interpreter makes a truly sophisticated LISP system available to S-100, CP/M users. It is capable of supporting serious AI efforts in such diverse fields as robotics, game playing, language translation, computer algebra, and theorem proving. Fully integrated into CP/M, it features infinite precision arithmetic, flexible program control constructs, an efficient garbage collector, & informative error messages. Most important for serious applications, it uses the most modern techniques to achieve extremely fast execution speeds. Please write The Soft Warehouse for details. We require a License Agreement be signed prior to shipment.
Release: Now
Price: $190
Included with price: On diskette: muLISP-79 COM file, Utility library file, Trace facility file, Pretty printer file, & a demo game program. Printed: 60 page Reference Manual, fully indexed.
Author: Albert D. Rich
Where to purchase it:

The Soft Warehouse
P.O. Box 11174
Honolulu, HI 96828

Program Name: Plotter Graphics Package
Hardware System: 8080/Z80 CP/M with either Houston Instruments Hiplot or Tektronix 40xx series terminal
Minimum Memory Size: Depends on how many routines are used
Language: Microsoft FORTRAN-80 and MACRO-80
Description: Set of FORTRAN callable subroutines which implement the standard CALCOMP plot routines: PLOTS, PLOT, FACTOR, WHERE, SCALE, LINE, SYMBOL, NUMBER and AXIS. Also includes several additional routines to support log and semi-log plots, with optional grids. All plotting is done through one simple "driver" routine which may be developed for any particular plotter. Drivers currently exist for Houston Instrument Hiplot and Tektronix 40xx series terminal (or equivalent). Entire ASCII character set is supported by SYMBOL routine. Provided as a 'User Library' from which externals may be satisfied at link time. Source code for both drivers are included. Several demonstration programs are included on the disk.
Release: Currently available
Price: Library and source for drivers; $100.00 Source for entire package: $1000.00
Included with price: 8" 3740 CP/M style diskette containing REL files for each routine, source for drivers, pre-built library for Hiplot (via SI02 port), and several sample programs using package. Enclosed manual describes calling arguments and operation of each routine. Coupon good for $100 off list price of Hiplot, from the Byte Shop of Columbia, S.C.
Author: Lawrence E. Hughes
Where to purchase it:

Mycroft Labs
P.o. Box 6045
Tallahassee, FL 32301



Directory, cont'd...

Program Name: BDS C Compiler
Hardware System: Anything supporting CP/M
Minimum Memory Size: 32K or more...the more, the better
Language: 8080 Machine code for 8080's and Z80's
Description: Compiles a good subset of UNIX C directly into relocatable load modules; a linker is provided to create the .COM files. Emphasis on speed and simplicity of compilation. The "C" language is aesthetic and concise—very powerful, yet relatively "low level," allowing the programmer to do just about anything. The compiler has been totally engineered to interact and co-exist with CP/M. It is comprised of two main segments, each about 10K, which operate in sequence to do a compilation. Instant support is always available by phone or mail from the author. No elaborate liscensing BS required.
Release: Currently available
Price: $125.00 ($20 for documentation alone).
Included with price: Compiler, Linker, Library Manager, Libraries containing over 75 utility and standard I/O functions, over 150K of sample sources, utility programs, a telecommunications program and more.
Author: Leor Zolman
Where to purchase it:

Lifeboat Associates
2248 Broadway
New York, NY 10024

Program Name: INFORMATION MASTER
Hardware System: 8080 or Z-80 with two or more disk drives
Minimum Memory Size: 32K
Language: CONVERS, a language similar to Forth and Stoic (note: no additional language package is required to run)
Description: Information Master is an information retrieval program for CP/M and CP/M compatible disk operating systems. The user creates free format text entries using his familiar text editor, setting off keywords or phrases with special character sequences. The program scans this text, creates a compact index, and builds a dictionary of all keywords encountered. Searches are made using single keywords or combinations of keywords in "and" and "or" clauses. A search of a data base with 500 entries typically takes about 12-15 seconds. After matches have been found, all or part of the original text is recovered for listing, viewing, or copying to a new disk file. Distributed on 8" single density floppy and some 5" formats, write for available formats.
Release: Now
Price: $37.50
Included with price: Information Master program with demonstration data base and configuration customizing program on disk. User's Manual.
Author: William B. Brogden
Where to purchase it:

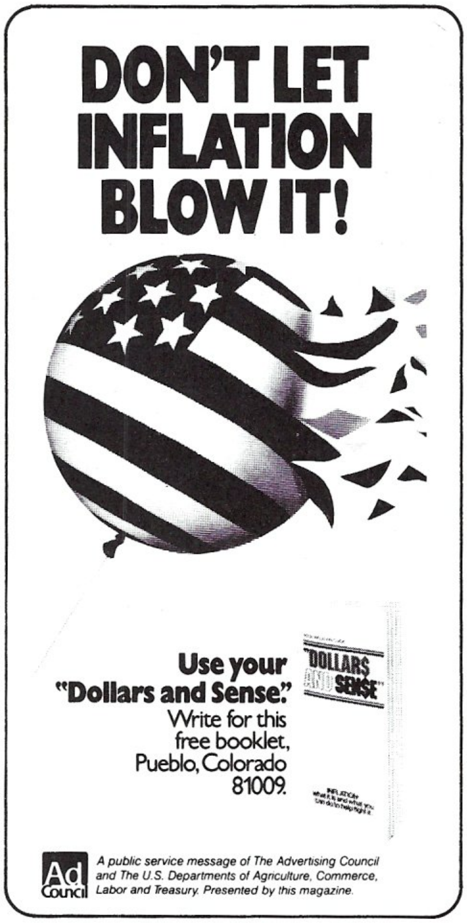
Island Cybernetics
P.O. Box 208
Port Aransas, TX 78373

Program Name: HAM Radio DX Package
Hardware System: 8080/Z80 8 inch CP/M
Minimum Memory Size: 24K
Language: Machine
Description: The Package provides operating information for the HAM DXer. This includes directions (compass heading), bearings (degrees), distance (miles, kilometers, hops), and time differential to the DX station. A paginated listing by prefix is produced. The programs are run on an interactive basis, simply by typing the COM file as a CP/M command. No RAM is taken by the data file as the data is called directly from the disk. The data base files can be edited to any length by the user.
Release: Now
Price: $22
Included with price: .COM files, 370 country data file, 50 state data file, improved directory utility. All on 8 inch disk.
Author: Ronald J. Finger
Where to purchase it:

FICOMP
3017 Talking Rock Drive
Fairfax, VA 22031

Program Name: Video ASM
Hardware System: Any memory mapped video board
Minimum Memory Size: 1K
Language: 8080 Assembler
Description: This video driver presents the ultimate in flexibility. The driver can be rommed if the user desires. It requires about 3/4K, and fits easily in a 2708 EPROM. The program will drive any size video board, with any line width or number of lines, without revision. The configuration and address of the video board are parameters provided at run time. It is quite capable of driving several different video boards, or several different windows on the same board, simultaneously. All parameters are stored in an 18-byte area. To drive multiple displays simultaneously, the user need only switch in or out the 18-byte parameter table desired. All control characters are stored in a second table. These are moved from the program body to a second table area, so they are subject to execution time revision by the user, even when the driver resides in ROM. When used in conjunction with an IMSAI VIO or Vectorgraphilc Flashwriter II, non-displayed 128 bytes of the VIO RAM to save all tables and variables. This driver offers such features as software scrolling, full cursor controls (up, down, left, right), screen clear, line erase, and user definable cursor character. It can be called with a single byte of data to be displayed, the address of a string to be displayed, or the address of a string to be displayed some variable number of times (repeat). The video driver will protect the contents of all registers during every call. They will be returned with their original contents.
Release: Currently available
Price: $40.00
Included with price: Program source code and documentation.
Author: Hawkeye Grafix
Where to purchase it:

Hawkeye Grafix
23914 Mobile St.
Canoga Park, CA 91307



S-100 MICROSYSTEMS' Bugs

In the March/April issue (Vol 1/No 2) there were some errors in the article titled "North Star Topics." They are as follows:

1) On pages 10 and 12 the order of paragraphs is incorrect. On page 10, from the end of column 1 proceed to page 12, column 1, line 3 of the text. From the end of the listing on page 12, column 2 proceed to page 10, column 2, line 2. Lastly, from page 12, column 1, line 17 proceed to page 10, column 2, line 1.

2) On page 10, column 2, line 2 of the program listing there should be a space after "CALL." Thus it should read: "CALL PRTBLK."

3) On page 12, column 2, line: the ORG should be "24C3H" and not "243CH," as shown.

We would like to thank Charles Stevenson for calling the errors to our attention.

ADVERTISER INDEX

THE ONLY MAGAZINE BY AND FOR S-100 SYSTEM USERS!

S-100 MICROSYSTEMS™

At last there is a magazine written exclusively for S-100 system users. No other publication is devoted to supporting S-100 system users. No longer will you have to hunt through other magazines for an occasional S-100, CP/M* or PASCAL article. Now find it all in one publication. Find it in **S-100 MICROSYSTEMS.**

Every issue of **S-100 MICROSYSTEMS** brings you the latest in the S-100 world. **Articles on applications, tutorials, software development, letters to the editor, newsletter columns, book reviews, new products, etc.** Material to keep you on top of the ever changing microcomputer scene.

$2.00 U.S.A.
S-100 MICRO SYSTEMS
JAN/FEB 1980 VOL. 1/NO. 1
THE IEEE S-100 STANDARD HAS ARRIVED!
See Pages 18-44
Also in this Issue
An Introduction to CP/M* BY JAKE EPSTEIN Page 6
Modifying the SDS VDB-8024 BY JON BONDY Page 11
Computerized Bulletin Board Systems BY RUSSELL GORR Page 16
An 8080 Disassembler BY WILLIAM YARNALL Page 48
and more
Complete Table of Contents on Page 3

$2.00 U.S.A.
S-100 MICRO SYSTEMS
MAR/APR 1980 VOL. 1/NO. 2
LINEAR PROGRAMMING TECHNIQUES IN PASCAL
See Page 14
Also in this Issue
An Introduction to CP/M* BY JAKE EPSTEIN Page 28
NORTH STAR Topics BY RANDY REITZ Page 10
Addressing The Cursor BY LARRY STEIN Page 34
S-100 Bus — New Versus Old BY SOL LIBES Page 39
and more
Complete Table of Contents on Page 3

VOL. 1/NO. 3
SORTING PROGRAMS FOR MICROCOMPUTERS
See Pages 35-47
Also in this Issue
…duction to CP/M, Part 3 by Jake Epstein Page 12
…Topics by Randy Reitz Page 18
…ramming — Part 2 by William Yarnall Page 24
…the Cursor, Part 2 by Larry Stein Page 30
and more
Complete Table of Contents on Page 3

SOFTWARE
CP/M*
Assembler
BASIC
PASCAL
applications
and lots more

SYSTEMS
Cromemco
Intersystems
North Star
IMSAI
SOL
and lots more

HARDWARE
8 bit & 16 bit CPUs
interfacing
hardware mods
bulletin board systems
multiprocessors
and lots more

*TMK Digital Research

Edited by Sol Libes
Published every other month

USA	Canada, Mexico	Other Foreign (Air)
	THREE YEARS (18 issues)	
□ $23.00	□$32.00	□$65.00
	TWO YEARS (12 issues)	
□ $17.00	□ $23.00	□$45.00
	ONE YEAR (6 issues)	
□$9.50	□$12.50	□ $23.50

□ New □ Renewal
□ Payment Enclosed
□ Visa
□ MasterCard
□ American Express
Signature ____________________
Card No. ____________________
Expiration date ____________________

□ Please bill me ($1.00 billing fee will be added; foreign orders must be prepaid)

S-100 MICROSYSTEMS
P.O. Box 789-M, Morristown, NJ 07960

S-100 MICROSYSTEMS™
ORDER FORM

Name ____________________
Address ____________________
____
City ____________ State ________ Zip ________

BACK ISSUES

□ 1-1 Jan/Feb 1980 $5.00
□ 1-2 Mar/Apr 1980 $2.50
□ 1-3 May/Jun 1980 $2.50
□ 1-4 Jul/Aug 1980 $2.50
□ 1-5 Sep/Oct 1980 $2.50
□ 1-6 Nov/Dec 1980 $2.50

Postpaid in USA; add $1.00 per issue foreign postage. Subscriptions start the month following receipt of order. Subscriptions cannot start with earlier issues.

CompuPro S-100 Motherboards: Designed for the Future,

AVAILABLE NOW

You won't have to throw away these motherboards when you upgrade your system – they are specifically designed to handle the new generation of 5 to 10 MHz CPUs coming on line, as well as present day 2 and 4 MHz systems. Faraday shielding between all bus signal lines minimizes crosstalk; additionally, when signal lines cross each other on opposite sides of the board, they do so at a 90 degree angle to minimize any chance of stray coupling. You'd expect the company that pioneered active termination to include true active termination, but we've gone one step better by splitting the termination load between each end of every bus line. And you won't have to junk your present computer box with our new motherboards – all sizes fit Godbout, Vector, Imsai, TEI, and similar enclosures.

These high-performance motherboards are available in **"unkit"** form (edge connectors and termination resistors pre-soldered in place for easy assembly), or fully **assembled** and ready to go.

#CK-024 20 slot motherboard with edge connectors – unkit $174, assm $214
#CK-025 12 slot motherboard with edge connectors – unkit $129, assm $169
#CK-026 6 slot motherboard with edge connectors – unkit $89, assm $129

NOTE: Most **CompuPro** boards are available in **unkit** form (sockets, bypass caps pre-soldered in place), **assembled**, or qualified under the **Certified System Component** (CSC) high-reliability program (200 hour burn-in, more). CSC memory boards run at 8 MHz, are **guaranteed** to run with 6 MHz Z-80s, and draw even less power than standard models.

CAREFUL . . . NOT ALL S-100 CPU BOARDS ARE CREATED EQUAL!

You'll appreciate the extras that go into our CPU boards; take IEEE spec compatibility, for example. While others may claim compatibility, we meet **all** timing specs – and we'll be glad to send you timing diagrams for our CPUs to prove it (just include an SASE). You don't have to compromise on another "me-too" board . . . choose **CompuPro.**

THE ENHANCED/ADVANCED Z-80A S-100 CPU BOARD

Superior design in an IEEE-compatible board gives the power for future expansion as well as system flexibility. Includes all standard Z-80A features along with power on jump/clear, on-board fully maskable interrupts for interrupt-driven systems, selectable automatic wait state insertion, provision for adding up to 8K of on-board EPROM, 4 MHz operation, and IEEE compatible 16/24 bit extended addressing. **$225 unkit, $295 assm, $395 CSC.**

THE COMPUPRO "RAM" SERIES OF STATIC MEMORY

Recommended for commercial, industrial, and scientific applications. 4/5 MHz standard operation, no dynamic timing problems, meets all IEEE specifications, low power/high speed chips used throughout, extensive bypassing, careful thermal design.

S-100 STANDARD MEMORY	unkit	assm	CSC
8K RAM IIA	**$169**	**$189**	**$239**
16K RAM X-16	**$329**	**$379**	**$479**
24K RAM XX-24	**$449**	**$499**	**$599**
32K RAM X-32	**$599**	**$689**	**$789**
S-100 EXTENDED ADDRESSING MEMORY (16/24 address lines; addressable on 4K boundaries)			
16K RAM XIV	**$299**	**$349**	**$429**
S-100 BANK SELECT MEMORY (Cromemco etc. compatible; addressable on 4K boundaries)			
16K RAM XIIIA-16	**$349**	**$419**	**$519**
24K RAM XIIIA-24	**$479**	**$539**	**$649**
32K RAM XIIIA-32	**$649**	**$729**	**$849**
SBC/BLC MEMORY			
32K RAM XI	**n/a**	**n/a**	**$1050**

OTHER S-100 BUS PRODUCTS

Godbout Computer Enclosure	**$289 desktop, $329 rack mount**
Active Terminator Board	**$34.50 kit**
2708 EPROM Board (less EPROMs)	**$85 unkit**
Memory Manager Board	**$59 unkit, $85 assm, $100 CSC**
2S "Interfacer I" I/O Board	**$199 unkit, $249 assm, $324 CSC**
3P Plus S "Interfacer II" I/O Board	**$199 unkit, $249 assm, $324 CSC**
Mullen Extender Board	**$59 kit**
Mullen Relay/Opto-Isolator Control Board	**$129 kit, $179 assm**
Vector 8800V S-100 Prototyping Board	**$19.95**

TERMS: Cal res add tax. Allow 5% for shipping, excess refunded. VISA®/Mastercard® orders ($25 min) call **(415) 562-0636**, 24 hrs. COD OK with street address for UPS. Prices good through cover month of magazine.

NEW! S-100 DUAL PROCESSOR CPU BOARD

The **Dual Processor Board** is here . . . and CPU boards will never be the same again. 8088 CPU gives true 16 bit power with a standard 8 bit S-100 bus; an 8085 gives compatibility with CP/M and 8080 software. Accesses up to 16 megabytes of memory, meets all IEEE S-100 bus specifications, runs 8085 and 8086 code **in your existing mainframe** as well as Microsoft 8086 BASIC and Sorcim PASCAL/M™, runs at 5 MHz for speed as well as power, and is built to the same stringent standards that have established our leadership in S-100 bus components. **Introductory prices: $385 unkit, $495 assm, $595 CSC.**

8085 single processor version of above: **introductory prices $235 unkit, $325 assm, $425 CSC.**

SPECTRUM S-100 COLOR GRAPHICS BOARD

Includes 8K of IEEE-compatible static RAM; full duplex bidirectional parallel I/O port for keyboard, joystick, etc. interface; and 6847-based graphics generator that can display all 64 ASCII characters. 10 modes of operation, from alphanumeric/semi-graphics in 8 colors to ultra-dense 256 × 192 full graphics. 75 Ohm RS-170 line output and video output for use with FCC approved modulators. **Introductory prices: $339 unkit, $399 assm, $449 CSC.** Don't settle for black and white graphics or stripped-down color boards; specify the **CompuPro Spectrum.**

Want graphics software? **Sublogic's 2D Universal Graphics Interpreter** (normally $35) is yours for **$25** with any Spectrum board purchase.

16K DYNAMIC RAM SPECIAL: 8/$59!

Expand memory in **TRS-80* -I** and **-II**, as well as machines made by **Apple, Exidy, Heath H89,** newer **PETs,** etc. Low power, high speed (4 MHz). Add $3 for 2 dip shunts plus TRS-80* conversion instructions. Limited quantity. *TRS-80 is a trademark of the Tandy Corporation.

PASCAL/M™ + MEMORY SPECIAL

PASCAL – easy to learn, easy to apply – can give a microcomputer with CP/M more power than many minis. We supply a totally standard **Wirth PASCAL/M™ 8" diskette** by **Sorcim,** with manual, for **$150 with the purchase of any memory board.** Specify Z-80 or 8080/8085 version. PASCAL/M™ available separately for $175; manual available separately for $10.

COMING SOON!

We've got a new board coming up that's **so** versatile some of our people have nicknamed it the "smorgasboard": it includes (among other things) a real-time clock, interval timer, interrupt controllers, and math processor. We've also got a board in the works that greatly enhances the throughput and performance of multi-user (2 or more terminal) systems, by assuming a lot of the overhead functions normally handled by the main CPU. Look for more details on these useful and functional products in the months ahead, or check with finer computer stores for additional information on these and other **CompuPro** products.

CompuPro™ from GODBOUT ELECTRONICS

Bldg. 725, Oakland Airport, CA 94614